L20: CLOUD STARTER

ese516: IoT Edge Computing

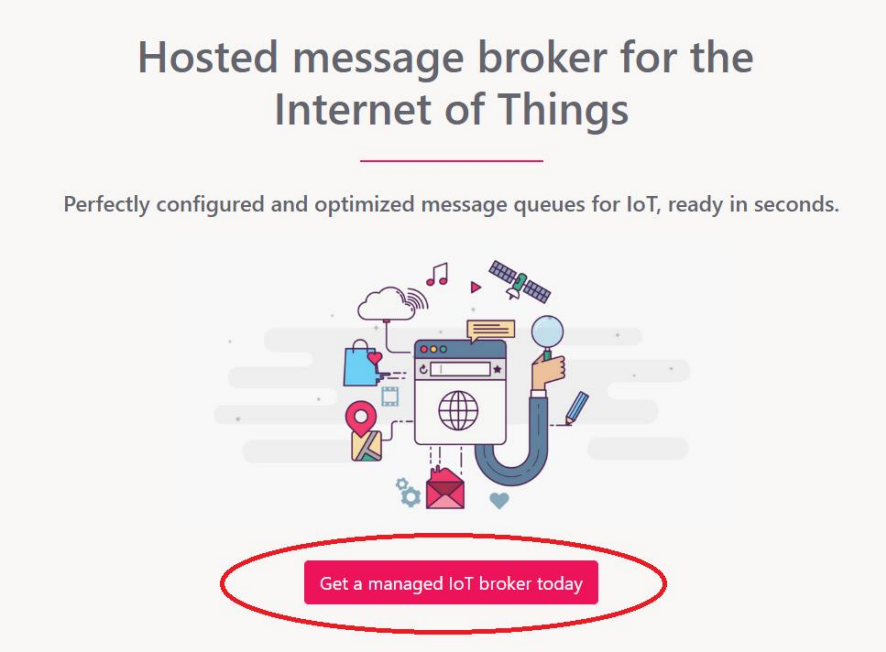
Monday, April 8, 2018

Eduardo Garcia edgarc@seas.upenn.edu

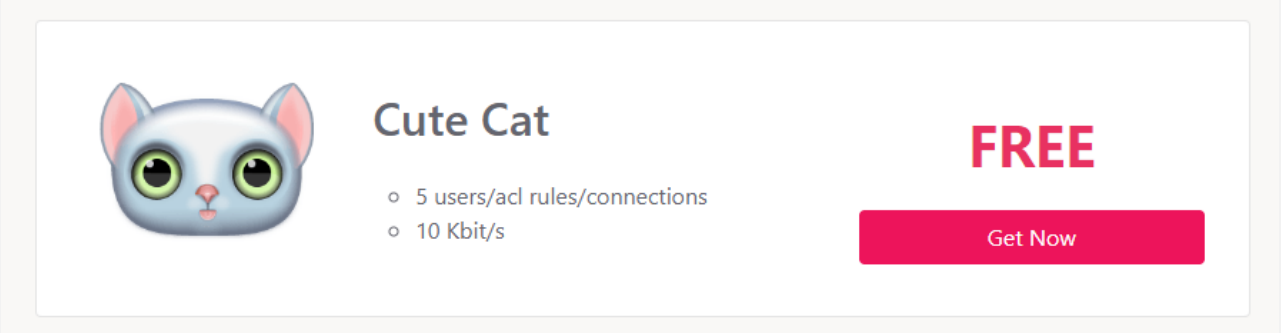
SETUP – MQTT BROKER

SETTING UP A MQTT BROKER

- The first step we will do is to setup an MQTT Broker. You can use any broker you like (such as Penn's one, at deet.seas.upenn.edu). We will use CloudMQTT for this example.

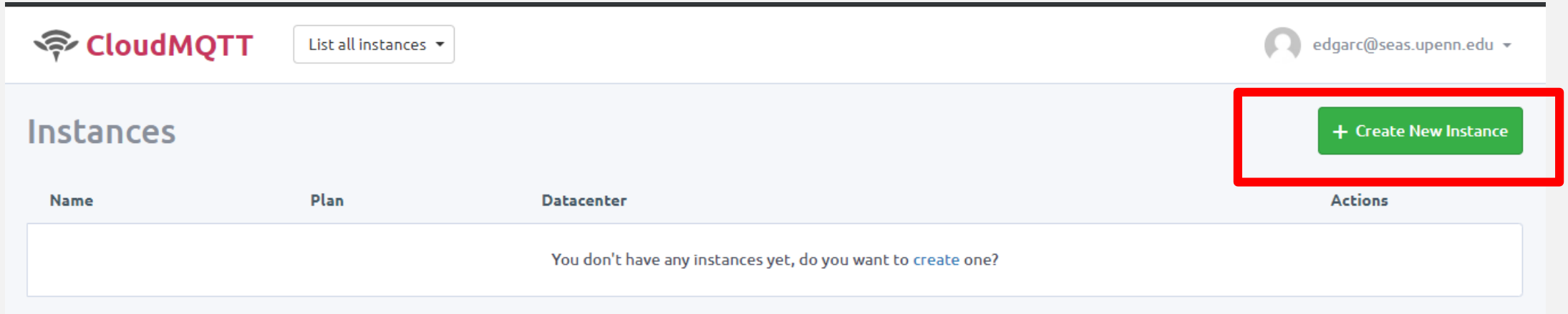


- Go to https://www.cloudmqtt.com/ and press on the "Get a Managed Iot Broker Today"
- Scroll down and select the Free Plan (Cute Cat)
- Make a new login.



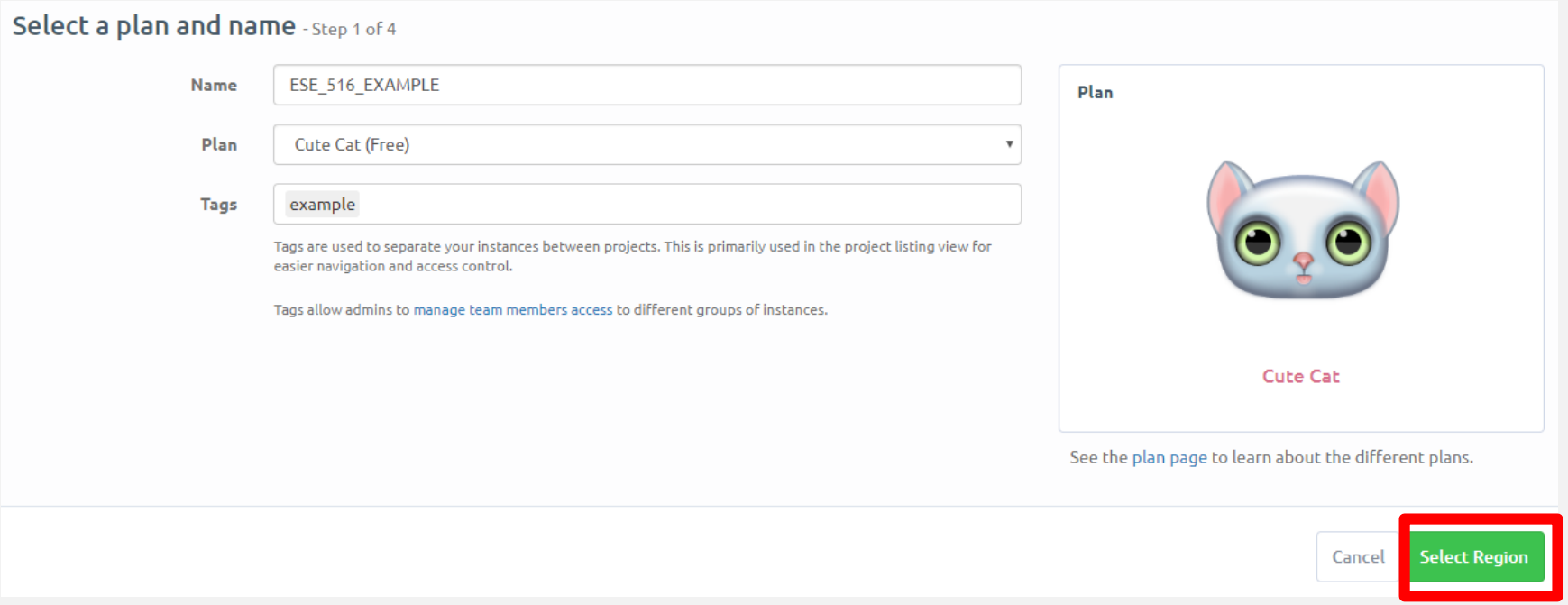
SETTING UP A MQTT BROKER

- Once logged in, create a new MQTT Instance



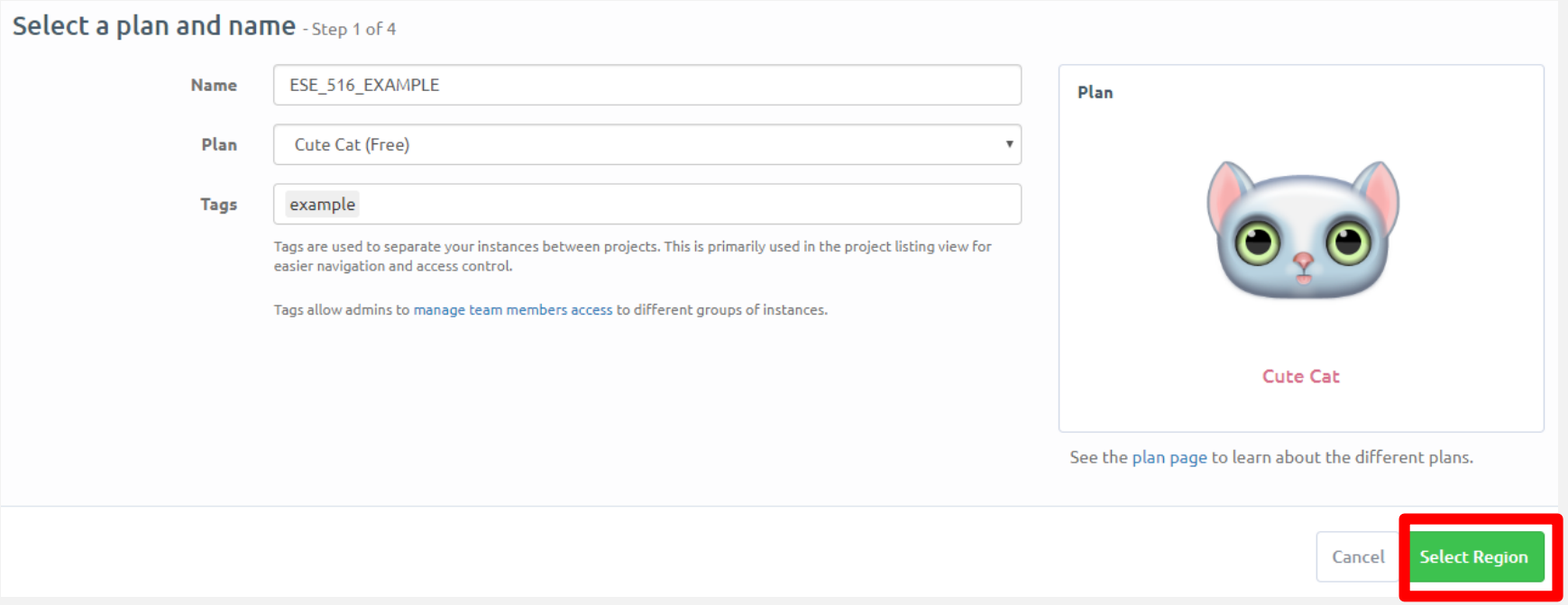
SETTING UP A MQTT BROKER

- Name your instance with a significant name and then hit Select Region



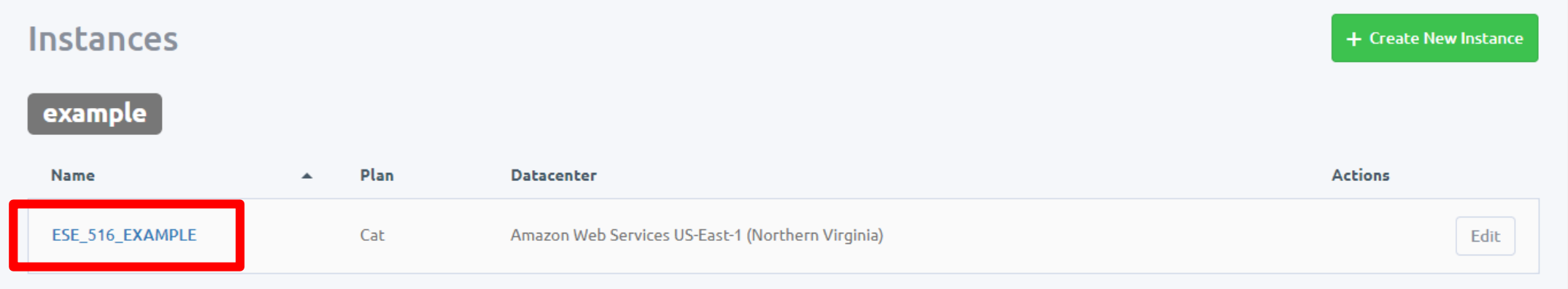
SETTING UP A MQTT BROKER

- Name your instance with a significant name and then hit continue through the options – select any region and create the instance.



SETTING UP A MQTT BROKER

- Once created, click on the name of the instance to access its data such as server address, user and password, etc. Keep this information for later use



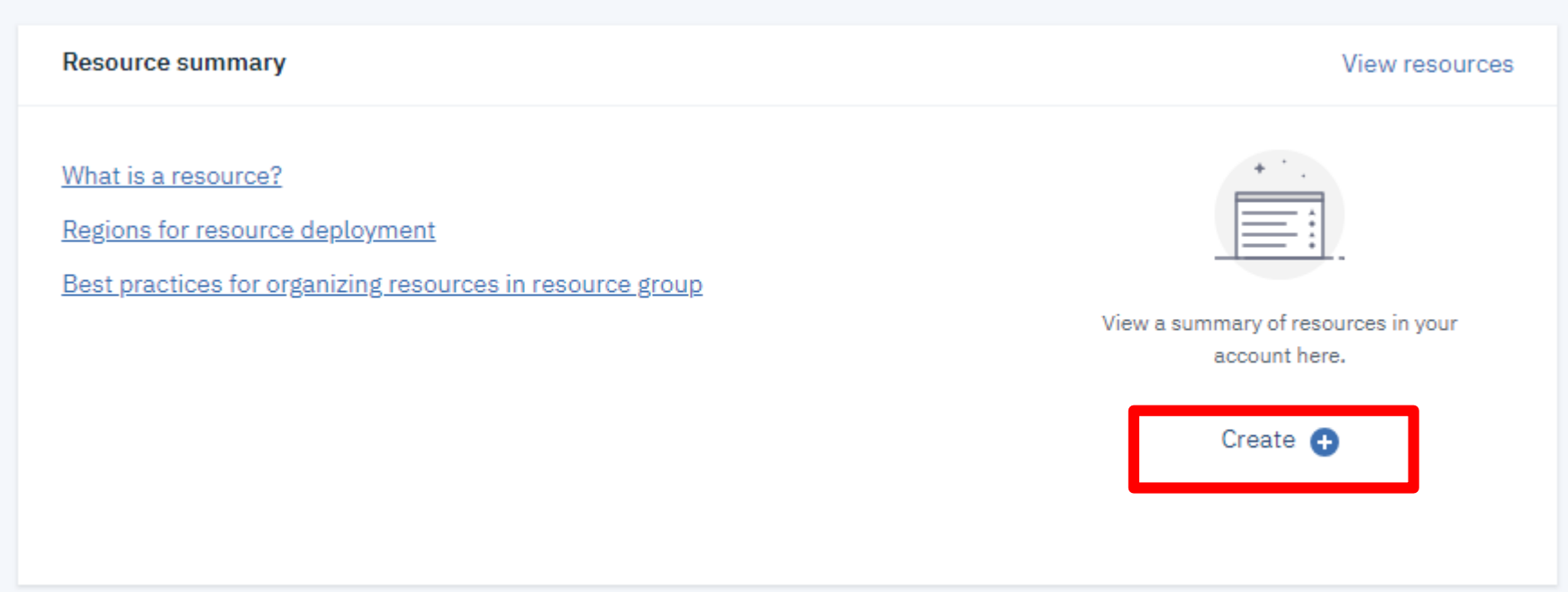
SETUP – IBM NODE RED

IBM CLOUD ACCOUNT

- Make an account at https://ibm.biz/BdZDGp

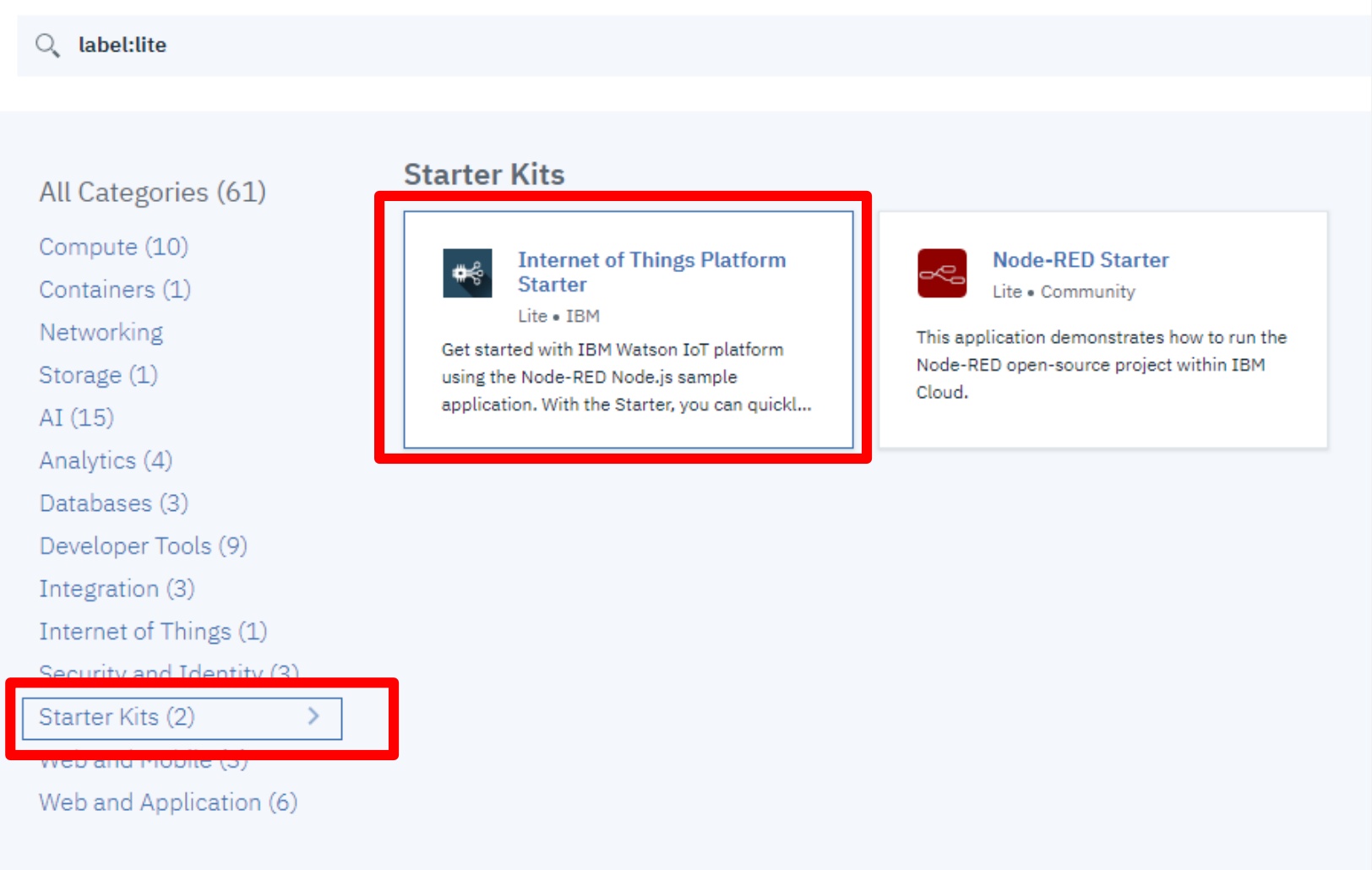
IBM CLOUD ACCOUNT

- Once you have your account ready, you will be greeted with the IBM Cloud Dashboard. Click on the "Create" under the "Resource Summary"



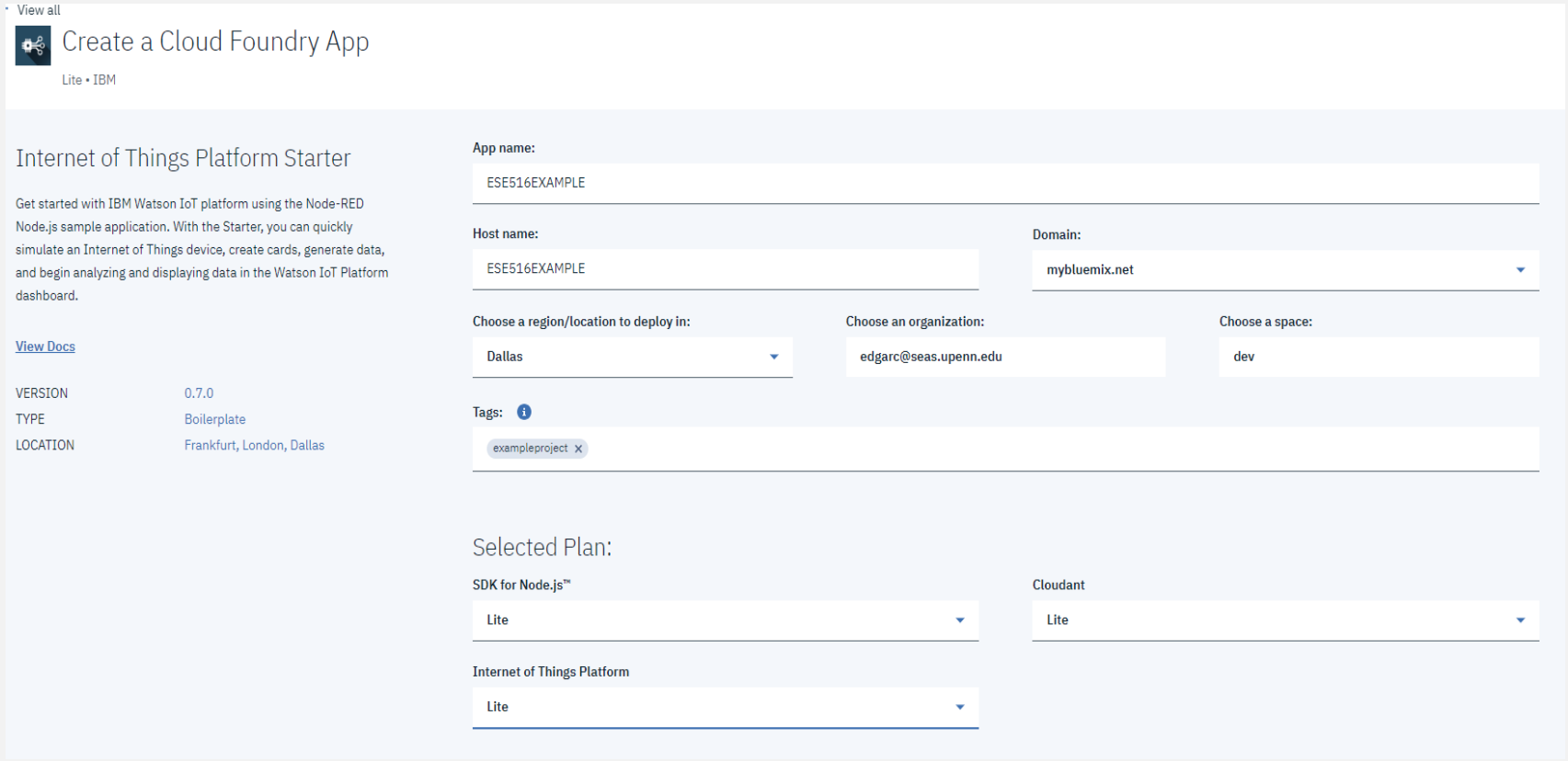
IBM CLOUD ACCOUNT

- Click on the "Starter Kits" and then click on the "Internet of Things Platform Starter".



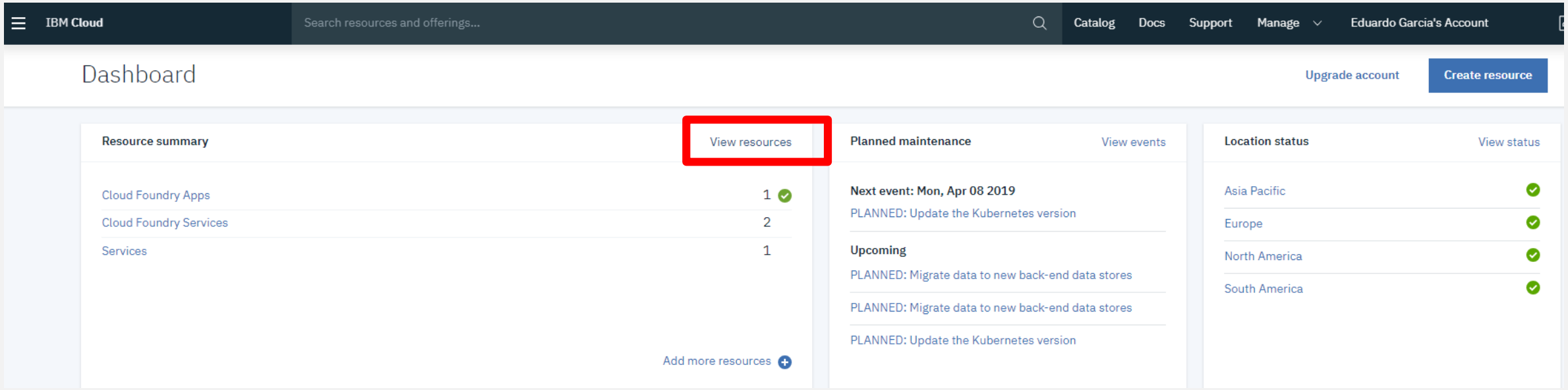
IBM CLOUD ACCOUNT

- Once you clicked, you will be greeted with a form to create an IoT boilerplate Platform Starter. Fill in the information with your project and hit Create.



IBM CLOUD ACCOUNT

- Once created, go back to your dashboard. Select "View Resources"



IBM CLOUD ACCOUNT

- Under "Cloud Foundry Apps" select the IoT Platform Starter App and click on it. This should take you to its main page.

Resource list

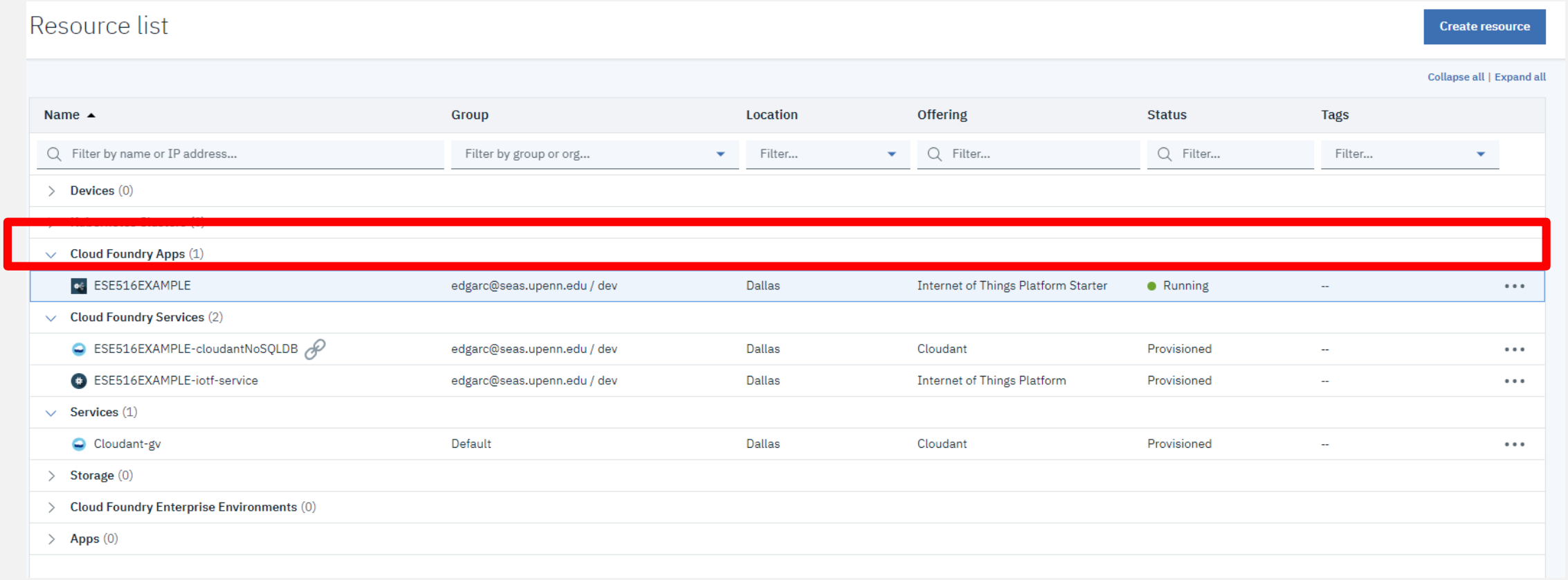
Create resource

Collapse all | Expand all

Name	Group	Location	Offering	Status	Tags
Filter by name or IP address...	Filter by group or org...	Filter...	Filter...	Filter...	Filter...
Devices (0)					
Cloud Foundry Apps (1)					
ESE516EXAMPLE	edgarc@seas.upenn.edu / dev	Dallas	Internet of Things Platform Starter	Running	--
Cloud Foundry Services (2)					
ESE516EXAMPLE-cloudantNoSQLDB	edgarc@seas.upenn.edu / dev	Dallas	Cloudant	Provisioned	--
ESE516EXAMPLE-iotf-service	edgarc@seas.upenn.edu / dev	Dallas	Internet of Things Platform	Provisioned	--
Services (1)					
Cloudant-gv	Default	Dallas	Cloudant	Provisioned	--
Storage (0)					
Cloud Foundry Enterprise Environments (0)					
Apps (0)					

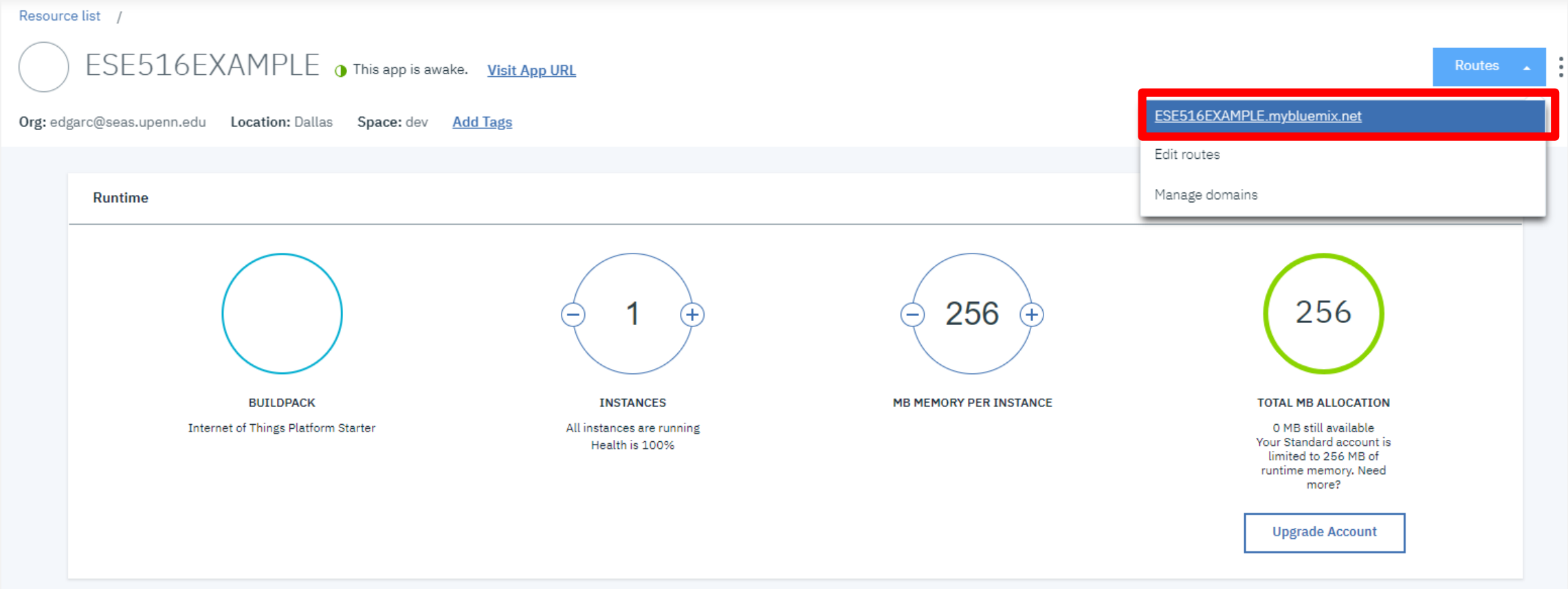
IBM CLOUD ACCOUNT

- Under "Cloud Foundry Apps" select the IoT Platform Starter App and click on its name. This should take you to its main page.



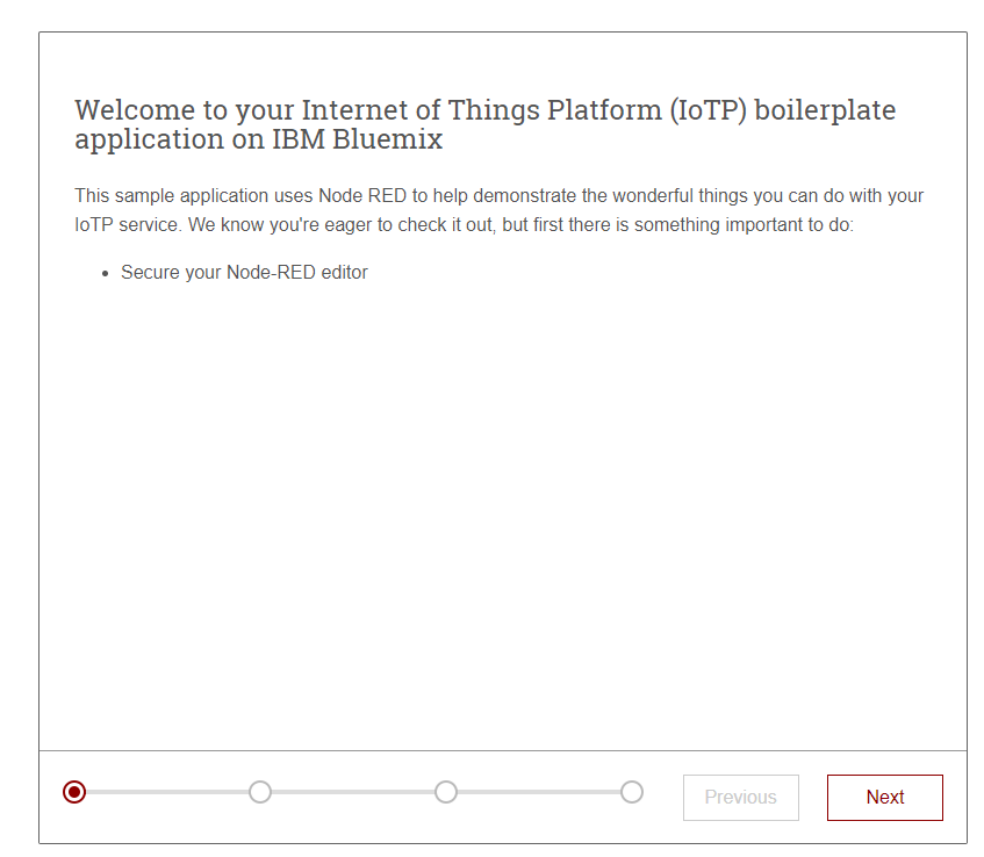
IBM CLOUD ACCOUNT

- Click "Routes" then click on the link that appears. This will take you to the main site of the App, Node-Red



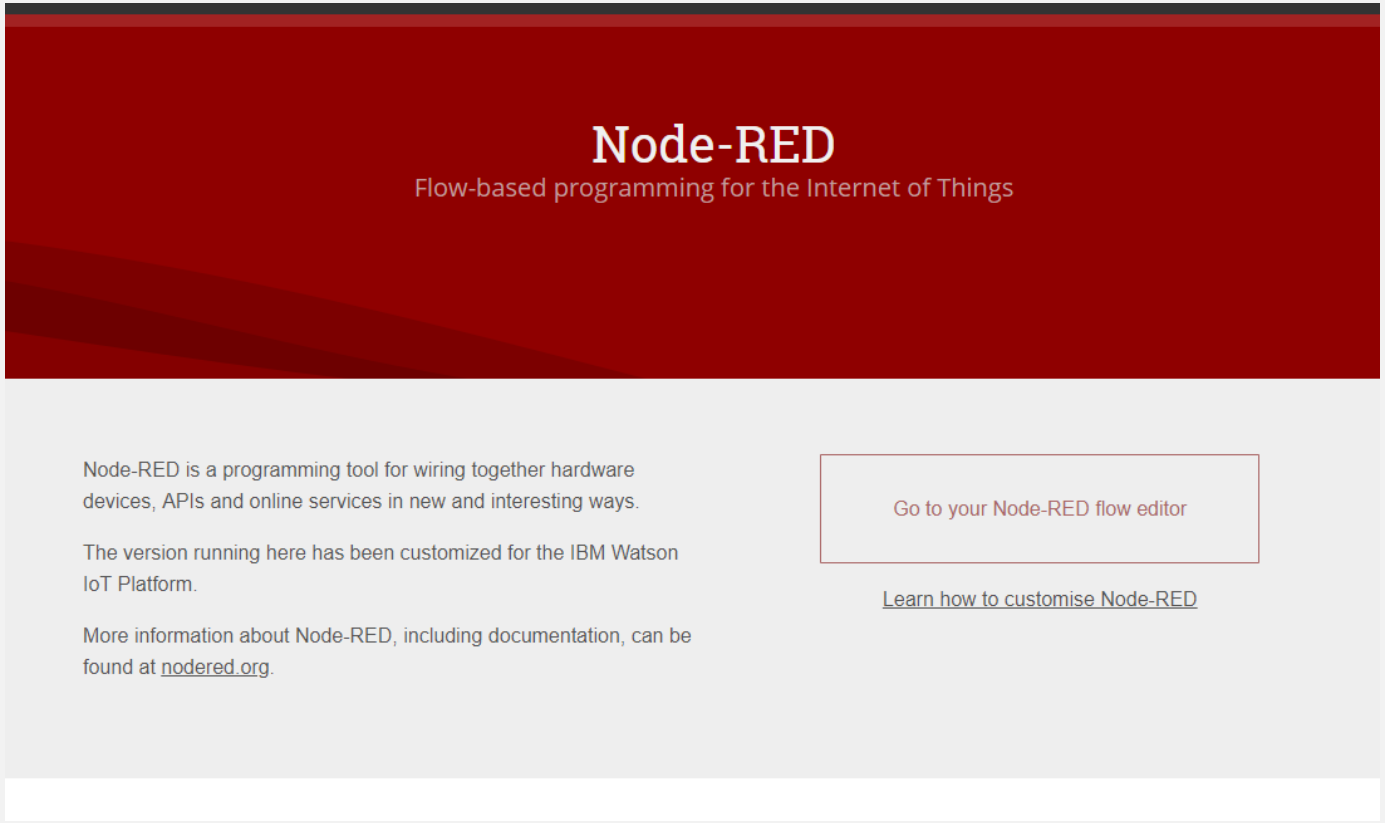
IBM CLOUD ACCOUNT

- When first opened, you will be asked to configure the Node-Red. You can add an username and password if you would like to protect your Node-Red instance. At the end, your Node-Red instance will be built.



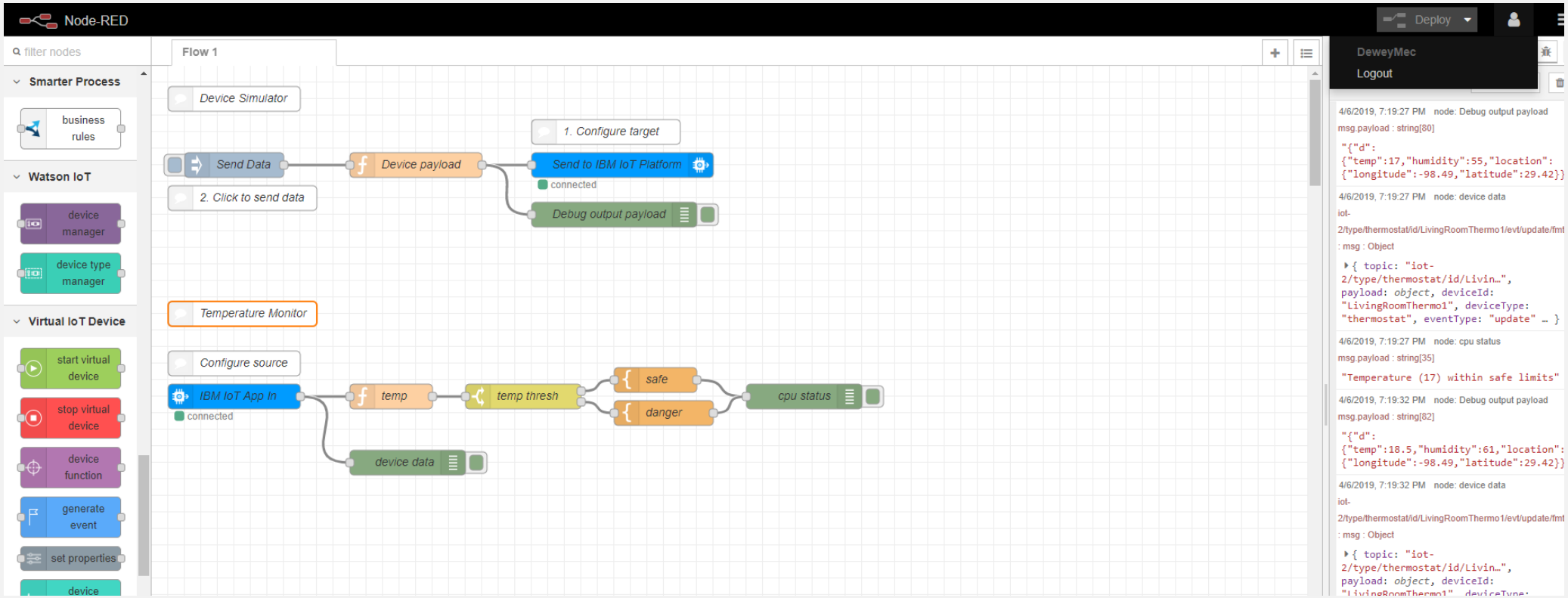
NODE-RED

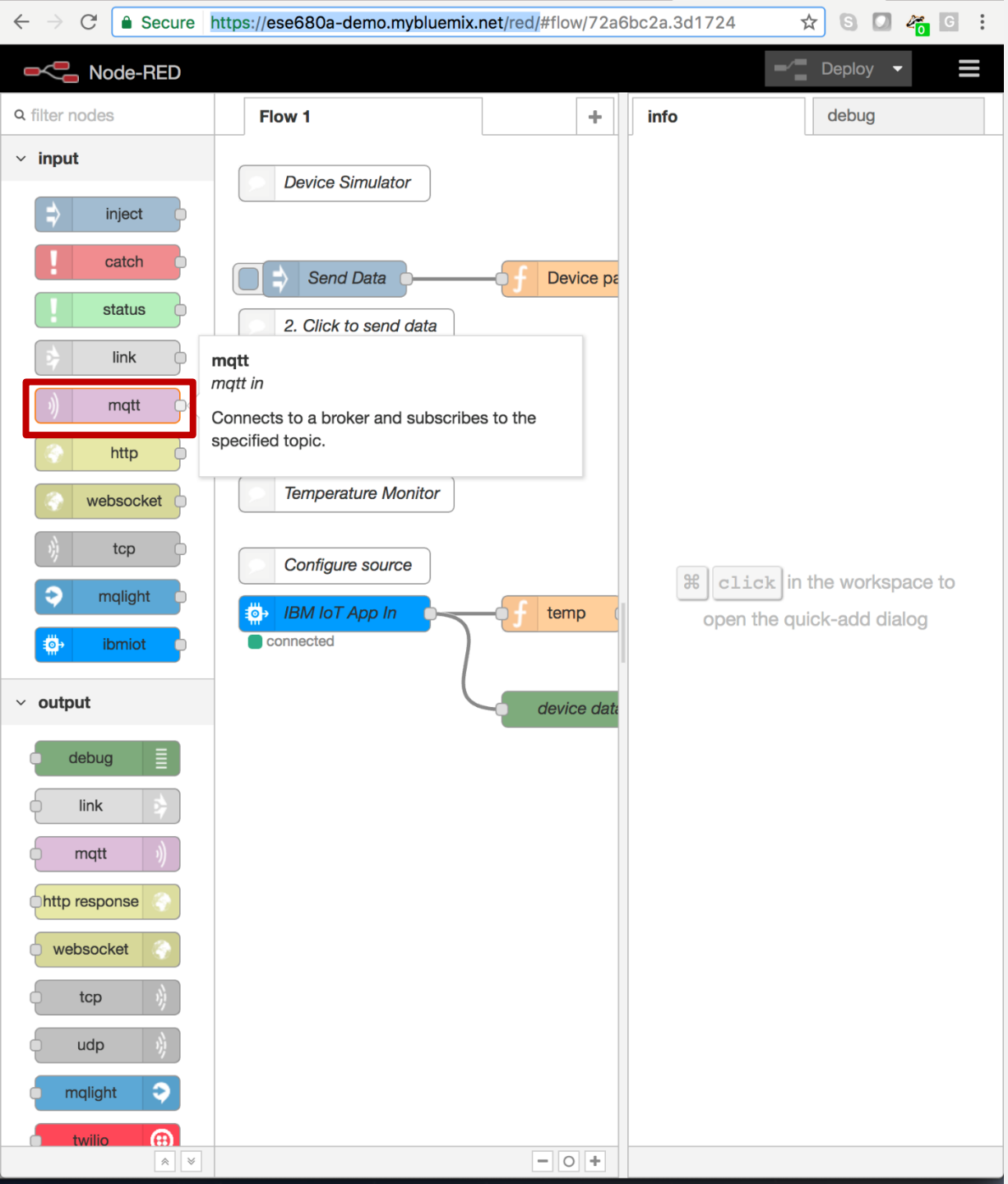
- Your Node-Red is now ready! Click on the “Go To your Node-RED flow editor” to open the editor. Otherwise, click on “Learn how to costumize Node RED” for a quick tutorial to start up.



NODE-RED

- You will be greeted with an example that simulates a device and captures data from said device and does something with it. We will modify this example a little bit. Before continuing, be sure to login with you recently made credentials, if you made them!



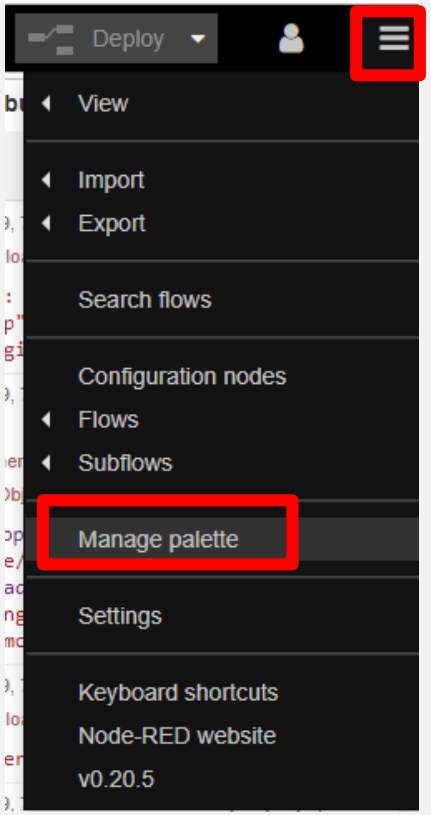


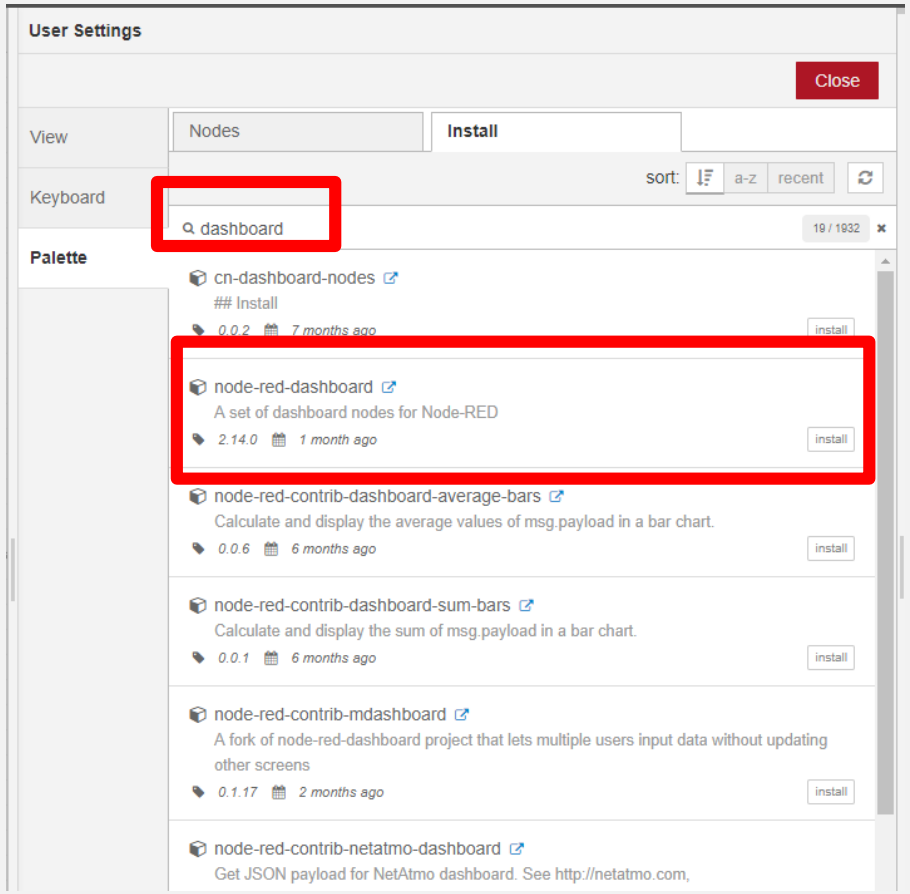
The Node-Red backend. Where the magic happens.

Note that MQTT comes baked in with the default Node-Red instance.

NODE-RED

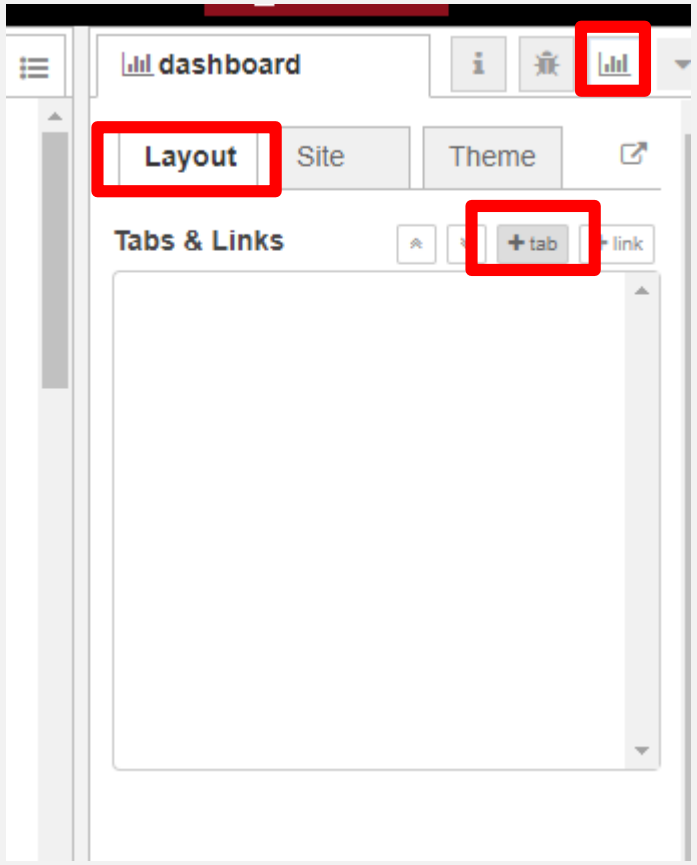
- Before continuing we need to install the libraries that will allow us to do a frond end. Hit the Menu button then Manage Palette. Then, search for "dashboard" under the "install" tab and choose to install the "node-red dashboard" Palette.





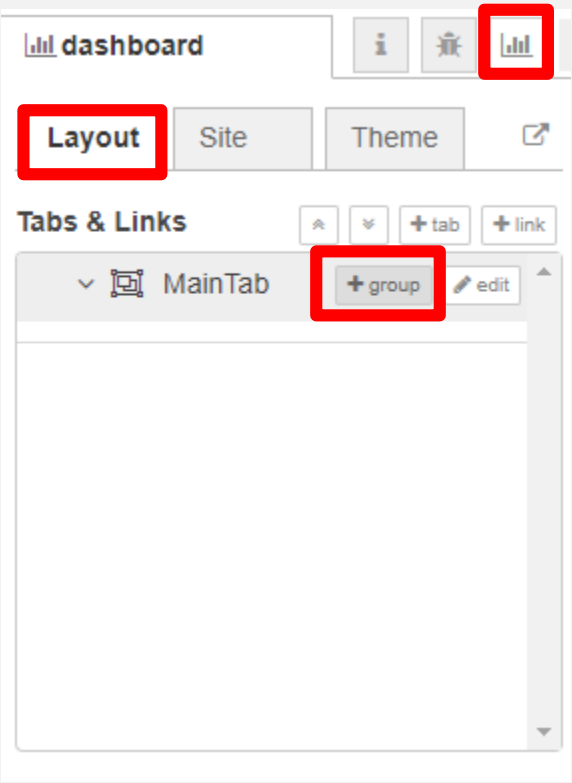
NODE-RED

- Now with the palette installed, you will have a third button appear. This will be the dashboard button. Click on it, and then under the Layout tab choose to add a new tab. Make a new tab and edit its name to something meaningful.



NODE-RED

- With a new Tab, we can add groups, which will be used to group together similar User Interface devices such as buttons. Add a new group to the device.

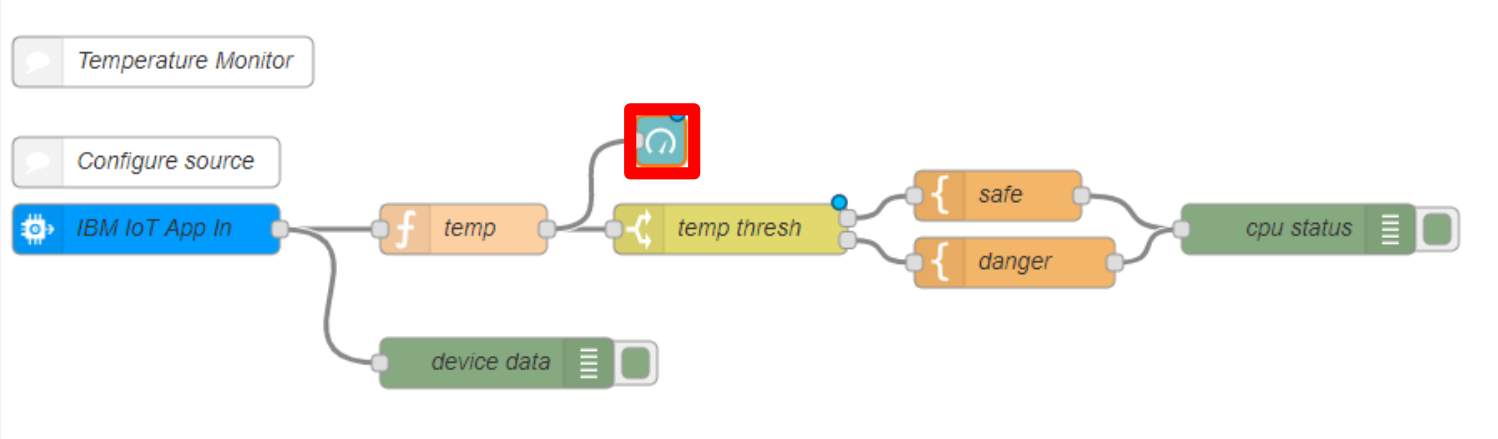


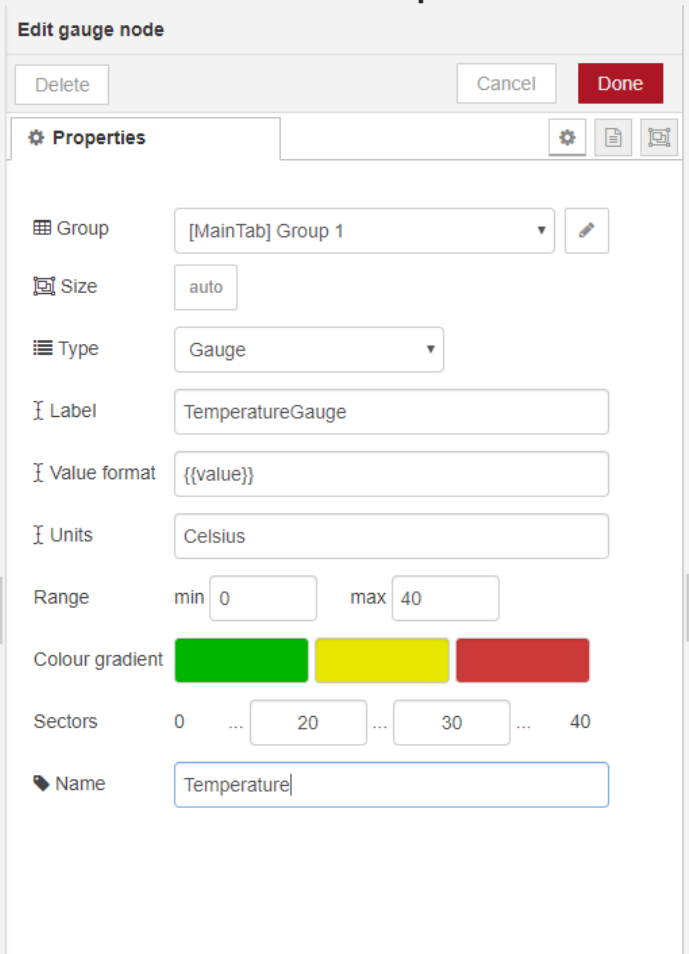
NODE-RED

- We will now make an edit so a simple button push is the one that simulates the device sending data.

NODE-RED

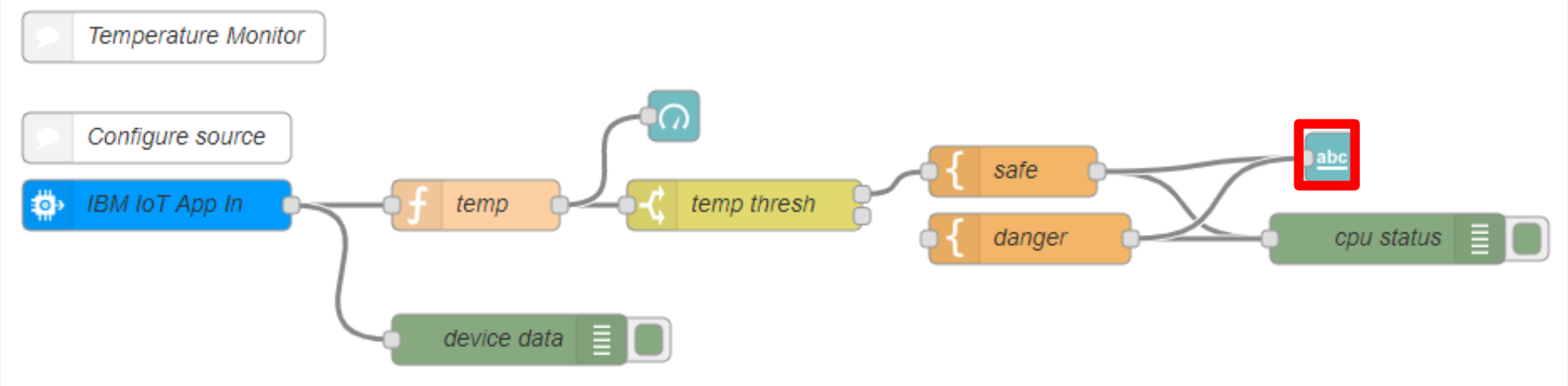
- Now we will edit the example a little bit. Add a "Gauge" from the "Dashboard" utilities and tie it to the output of the "Temp" JSON code. Double click and change the properties as shown below. This will help us simulate the data better!





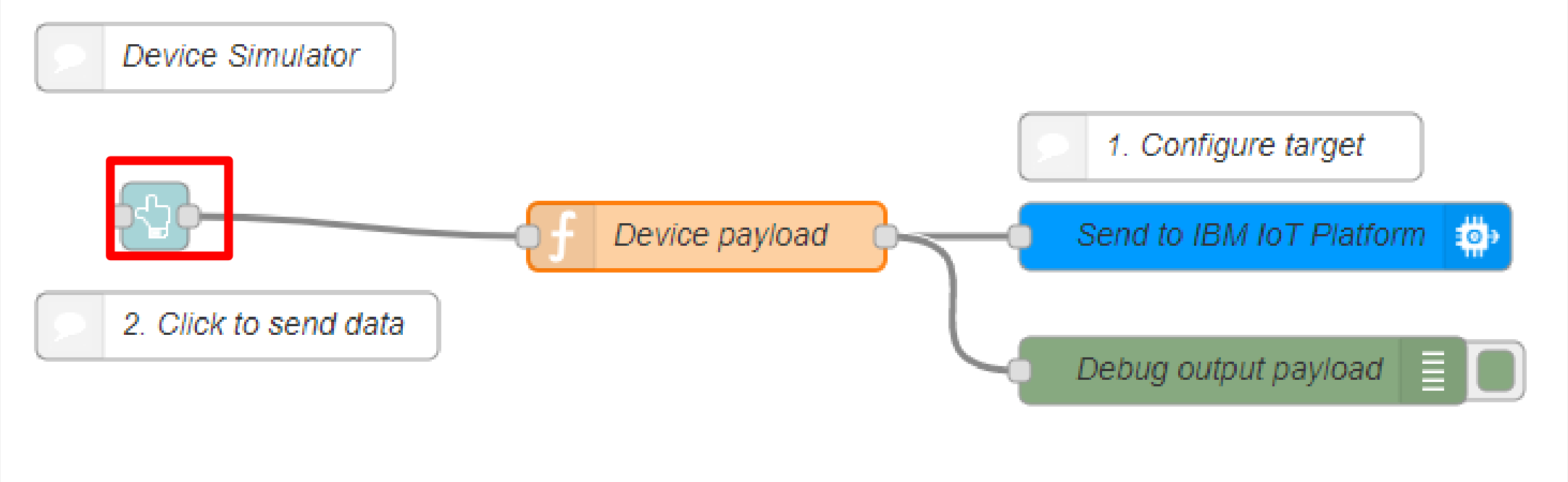
NODE-RED

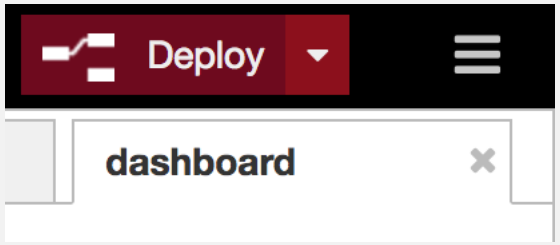
- Now, lets add a text area so we can read the data generates on the "safe" and "danger" JSON code. Add a "text" block and wire it as shown below.



NODE-RED

- Now on the Device Simulator side, let's get rid of the input block and use a button. For this, remove the first block of the chain and replace it with a "Button" block, like shown below.



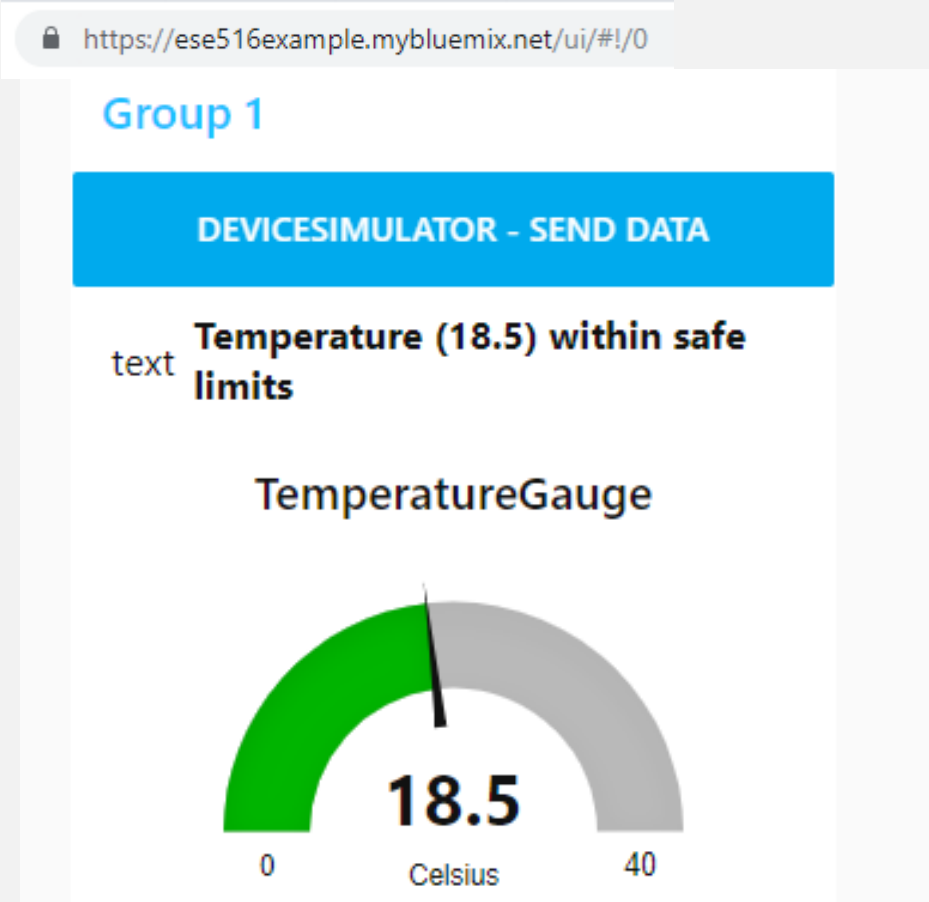


You'll have to **Deploy** before seeing your changes.

Explore the other **Deploy** options under the drop down arrow.

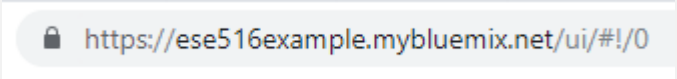
Now you'll be able to see something on the dashboard.

Press the button repeatedly – it will mock the sending of data!



[your base url]/ui now brings up a dashboard webpage.

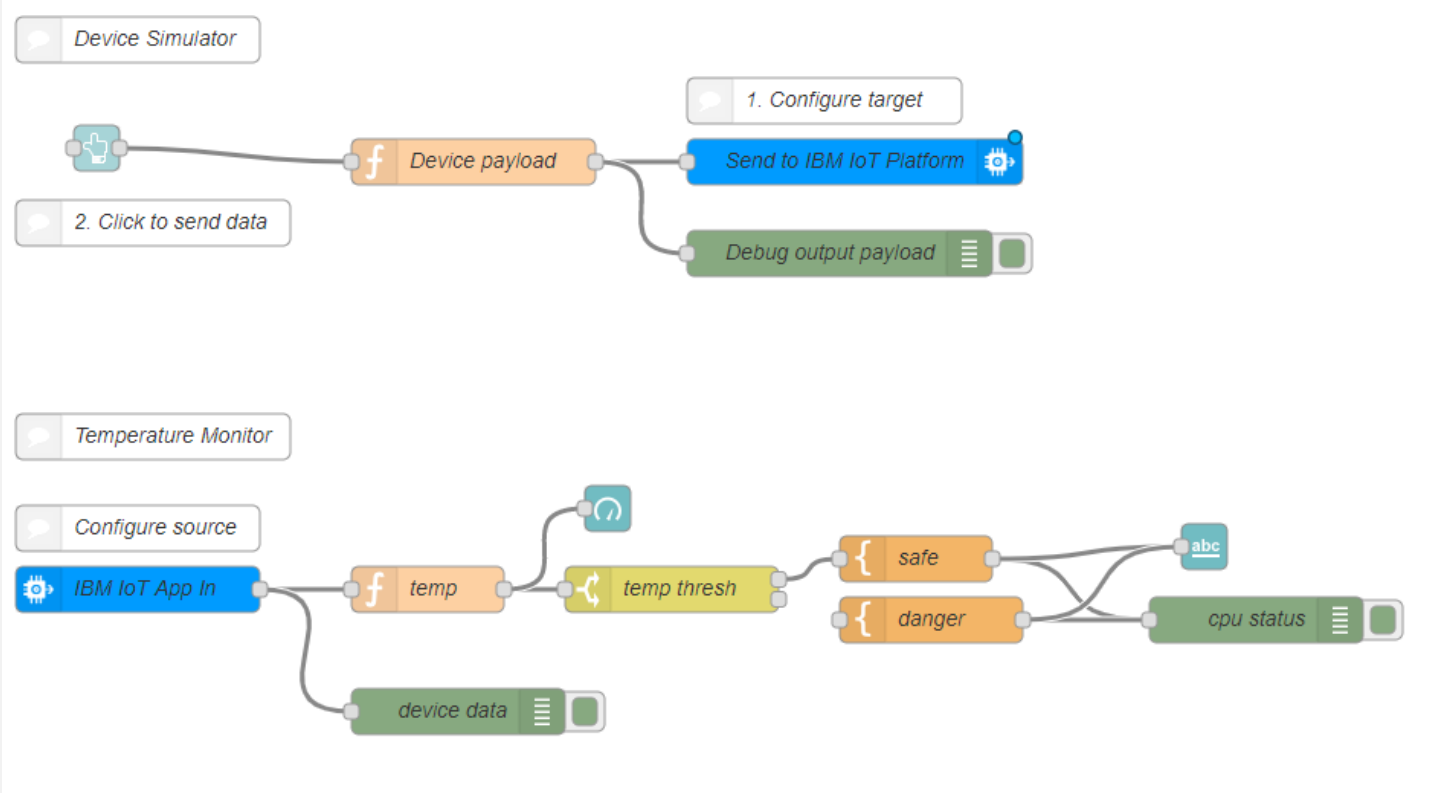
It's blank by default – we need to add some dashboard items to do something.



EDITING EXAMPLE TO USE OUR MQTT BROKER

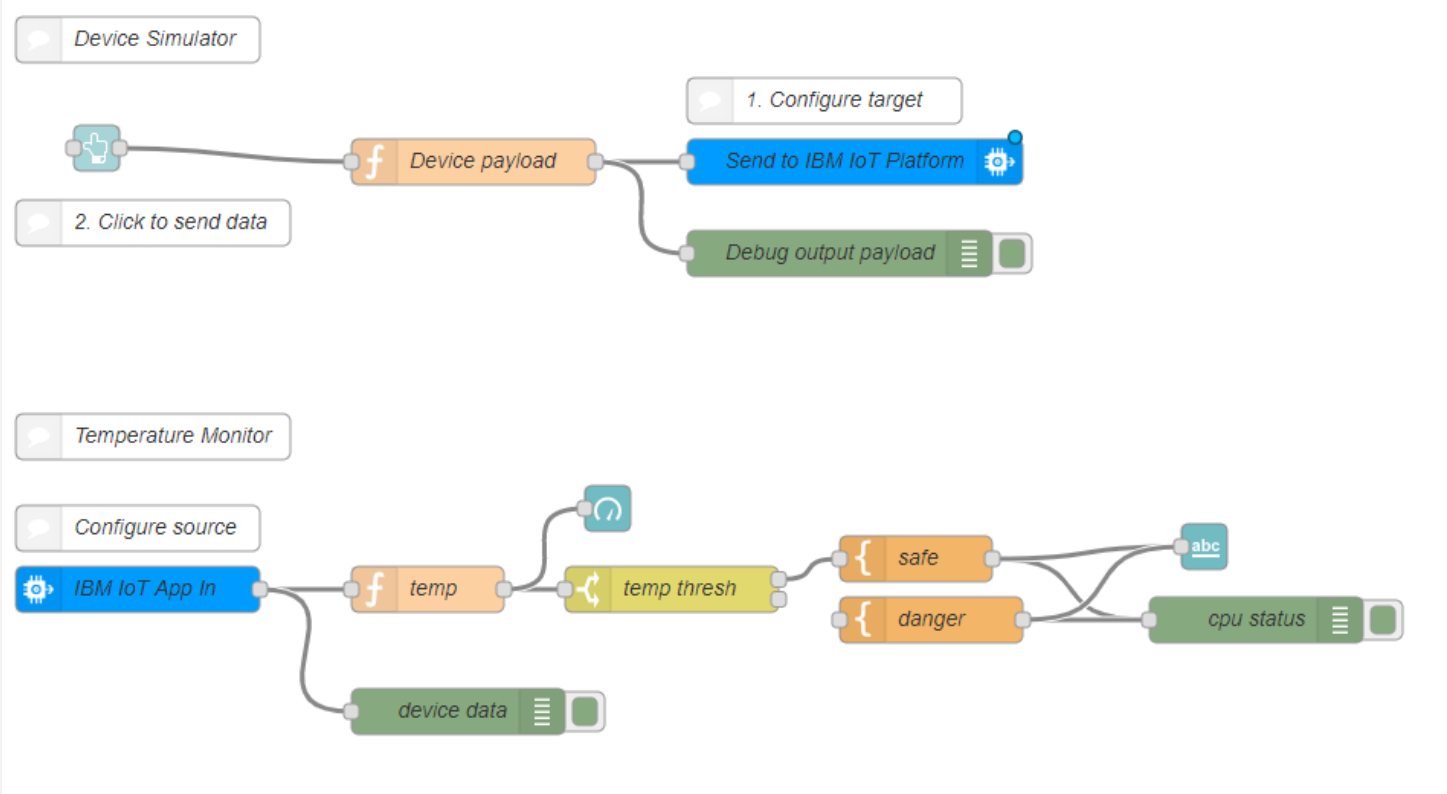
NODE-RED – USING OUR MQTT BROKER

- As you can see from the previous example, we are using IBMs IoT platform to send-receive data. That is great! However to make our projects easier, we will use our MQTT brokers as the communication layer. This is because adding an embedded device to the IBM IoT cloud can take time and is out of the scope of the class.



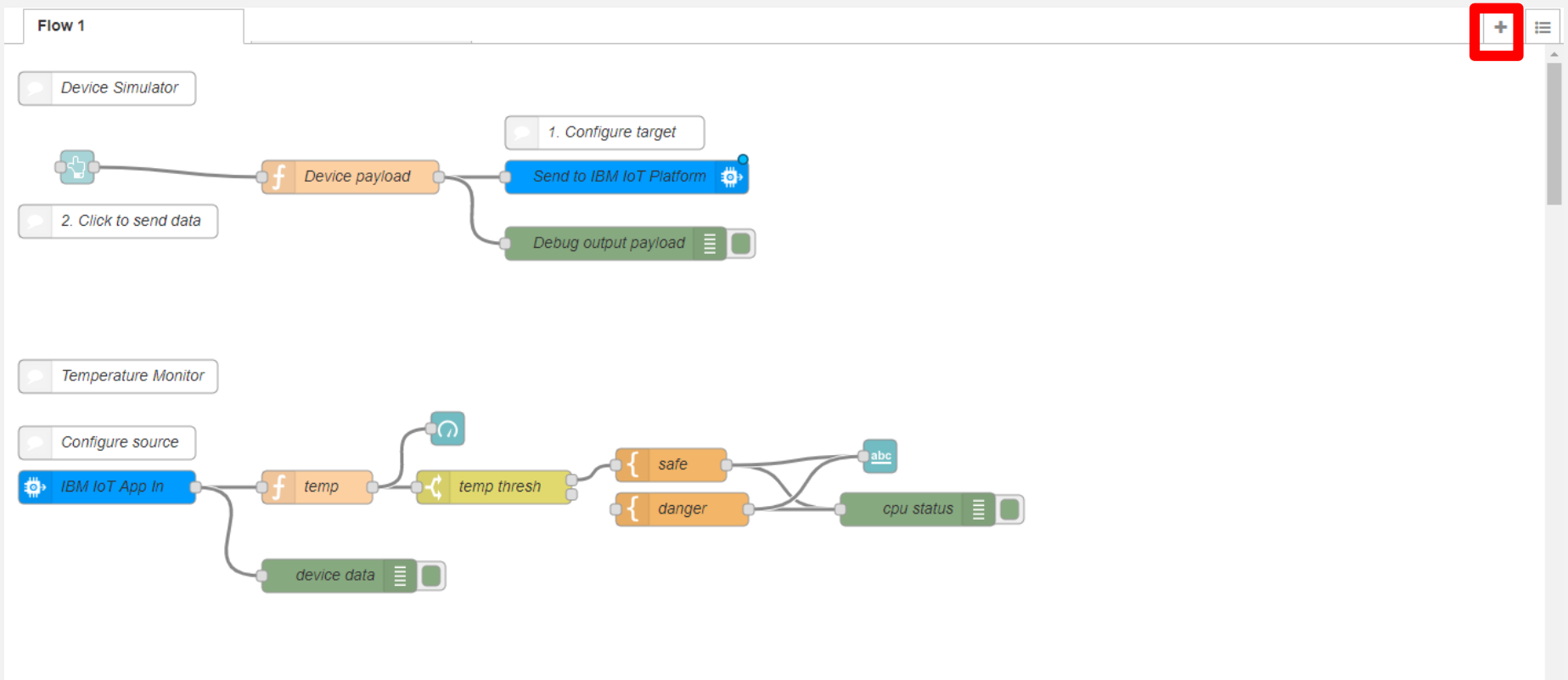
NODE-RED – USING OUR MQTT BROKER

- To not lose our example, we will create another tab, copy the flow, and edit that flow. **Please do not work over this workflow!** It is useful to have it just in case.



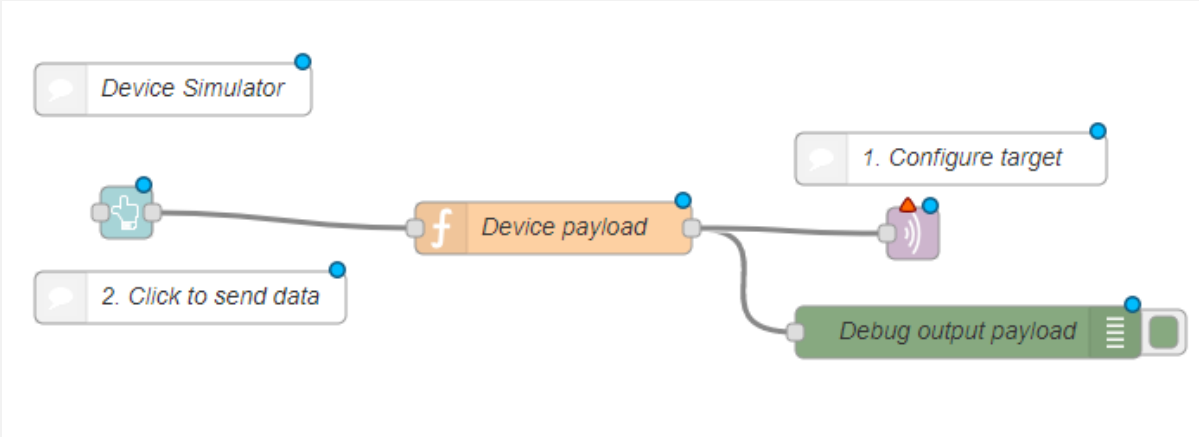
NODE-RED – USING OUR MQTT BROKER

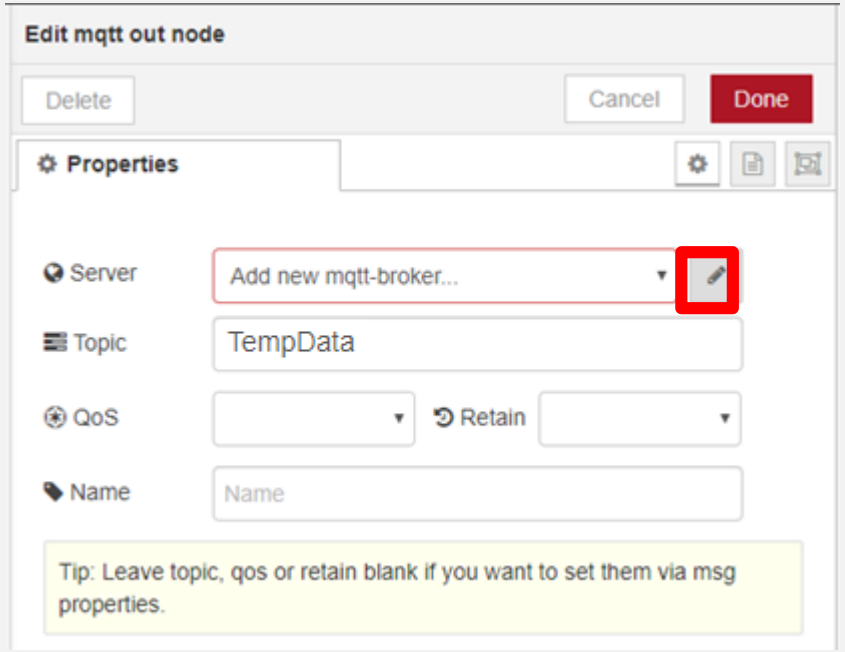
- Hit the Plus sign on top of our workspace. This will add a new Flow tab (Called Flow 2). Copy all the blocks from the Flow 1 tab and paste them on the Flow 2 tab. We will work on this Flow 2 tab.



NODE-RED – USING OUR MQTT BROKER

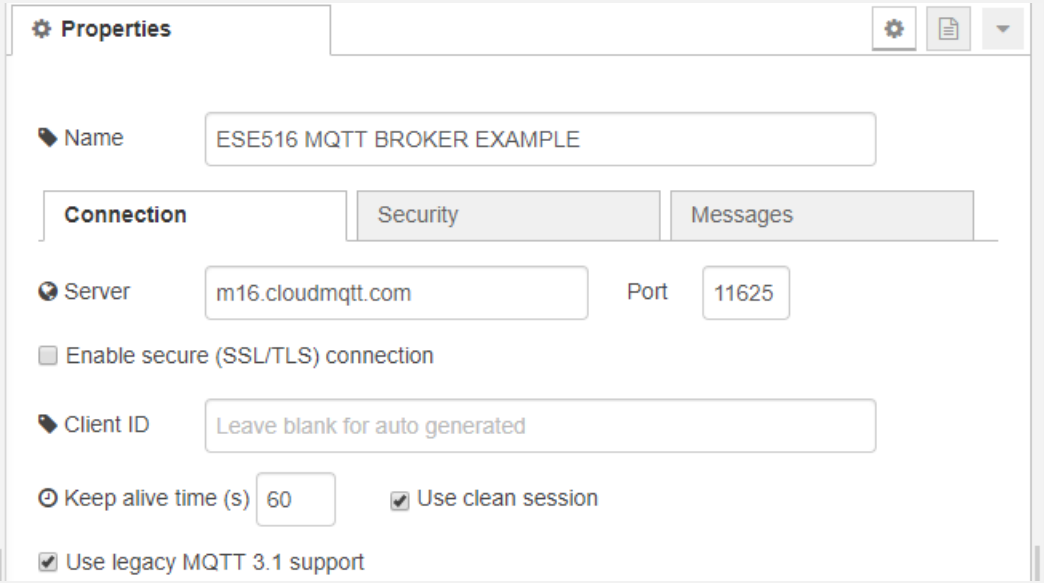
- Replace the output IBM IoT block with an MQTT output block as shown below. Set the Topic to TempData. Then, double click on the MQTT block. Click the edit button…and see the next slide.

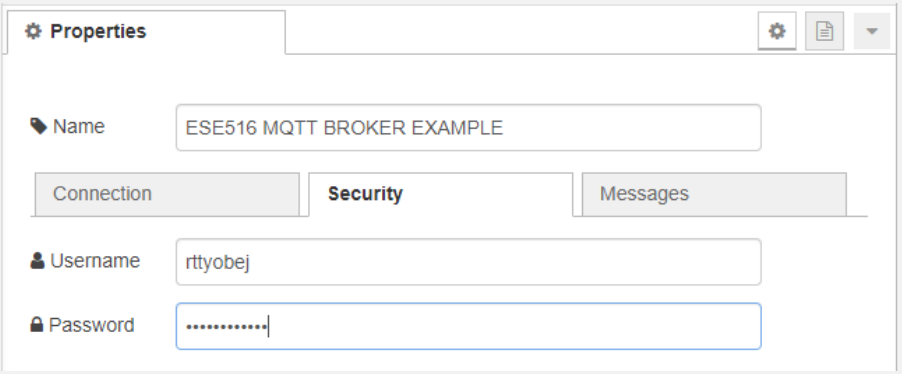


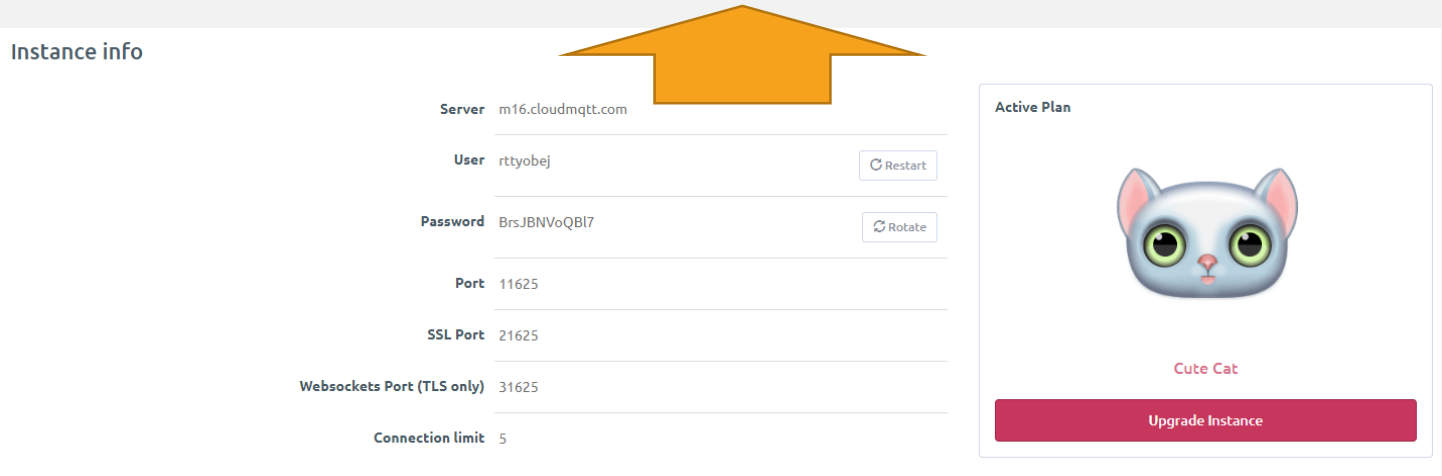


NODE-RED – USING OUR MQTT BROKER

- Put all the information from your CloudMQTT server you created beforehand. Add the server, Port, Username and Password,

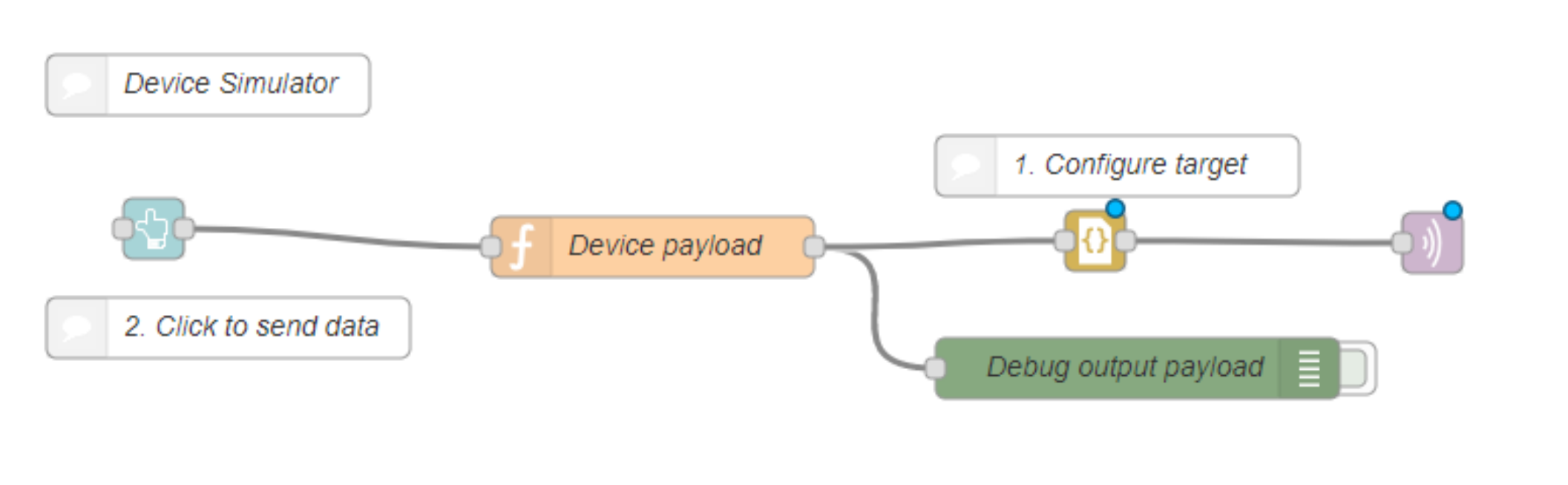






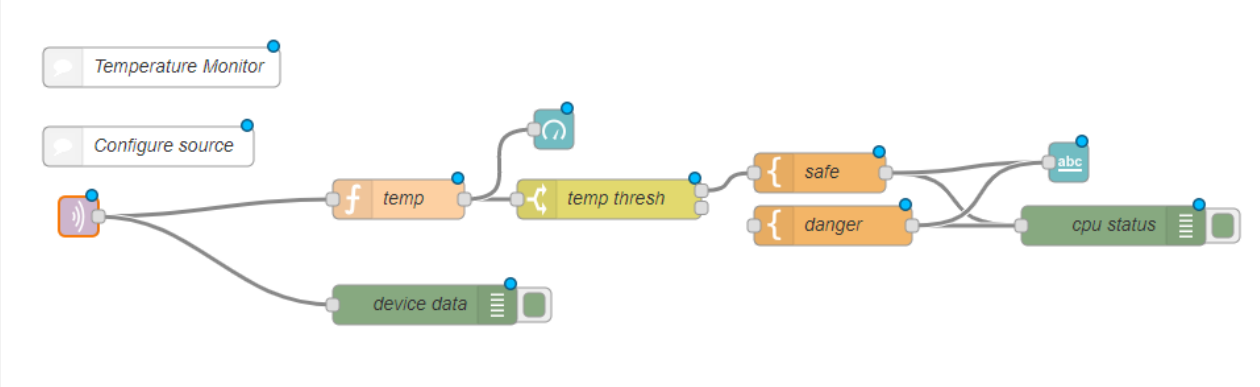
NODE-RED – USING OUR MQTT BROKER

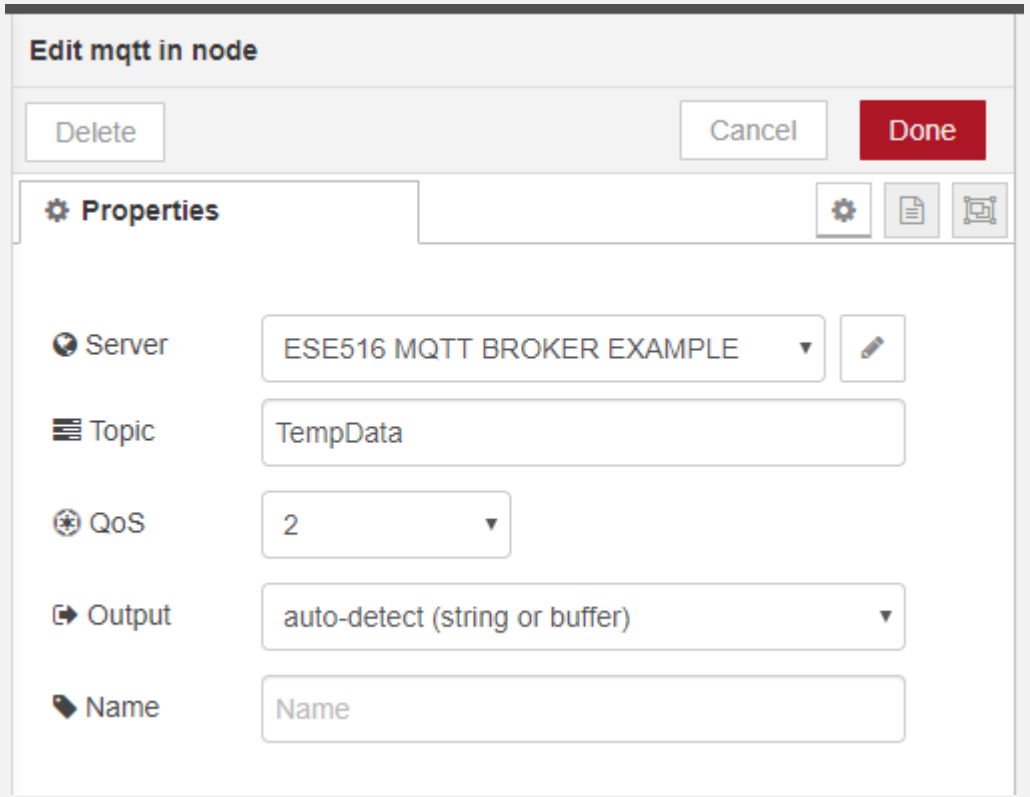
- Now, the IBM IoT block we removed did the job of parsing a string, generated at the "Device Payload" block (double click that node if you want to see the code). We will need to add a JSON block to parse the data correctly before sending it. Add a JSON block as shown below (it is under the "function" group.)



NODE-RED – USING OUR MQTT BROKER

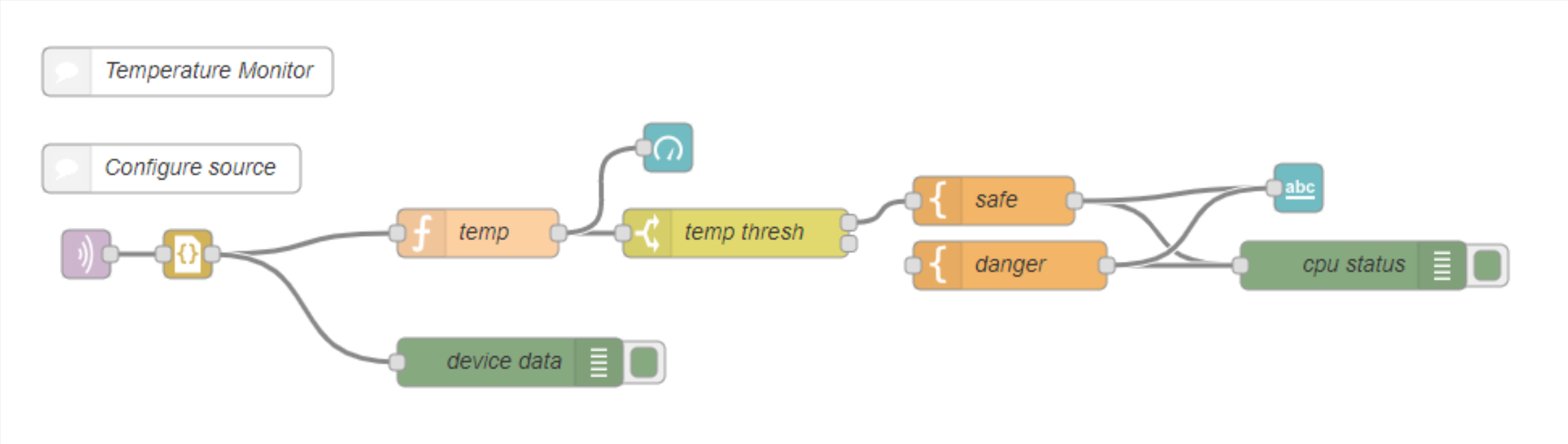
- Now, replace the INPUT IBM IoT block with an input MQTT block. Double click on it and choose the MQTT broker you already added (should have been added to the dropdown list once created) and choose the topic to listen to to be the same as the previous step, the "TempData" topic.





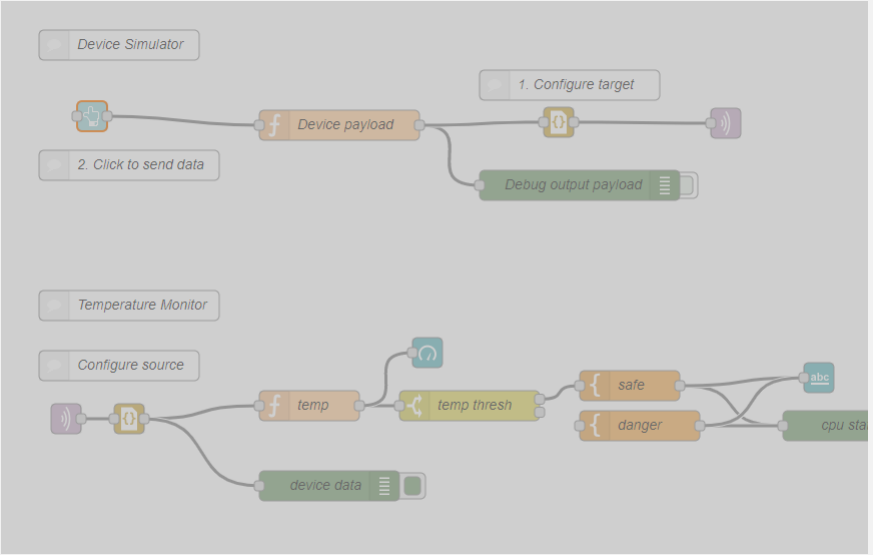
NODE-RED – USING OUR MQTT BROKER

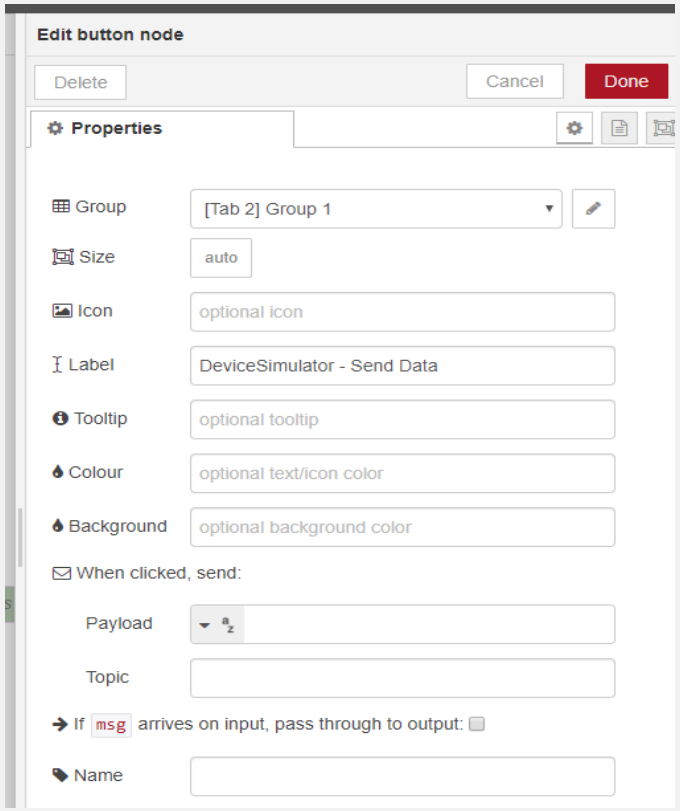
- As explained before, we will need to add a JSON parsed between the MQTT input and the "temp" block as shown below.

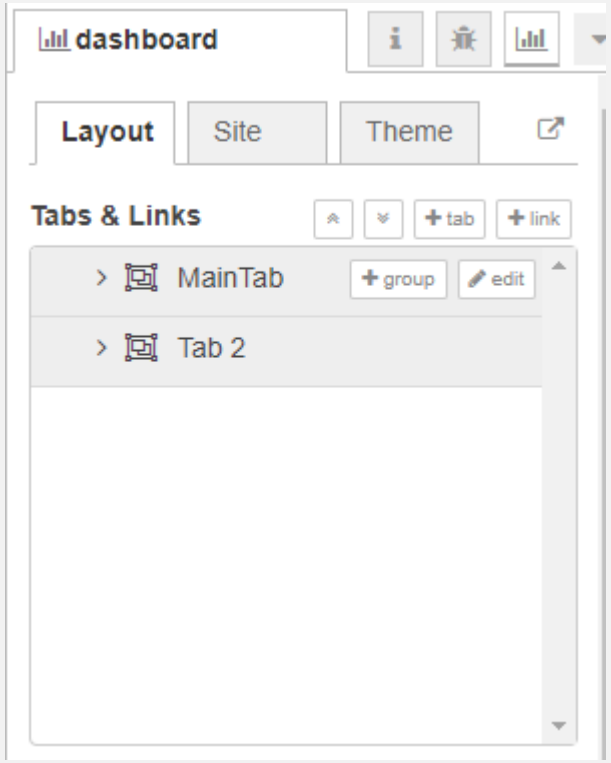


NODE-RED – USING OUR MQTT BROKER

- As shown previously, make a new tab on the Dashboard Layout, and edit the button, gauge and text of Flow 2 to appear on this new tab.

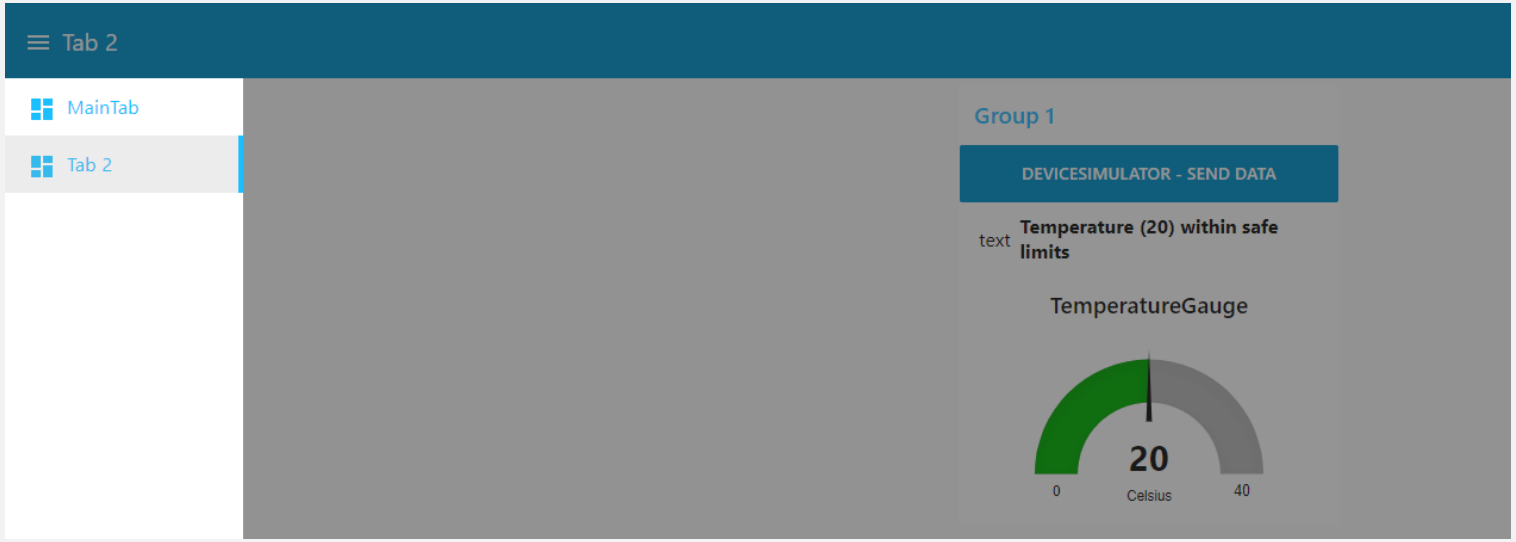






NODE-RED – USING OUR MQTT BROKER

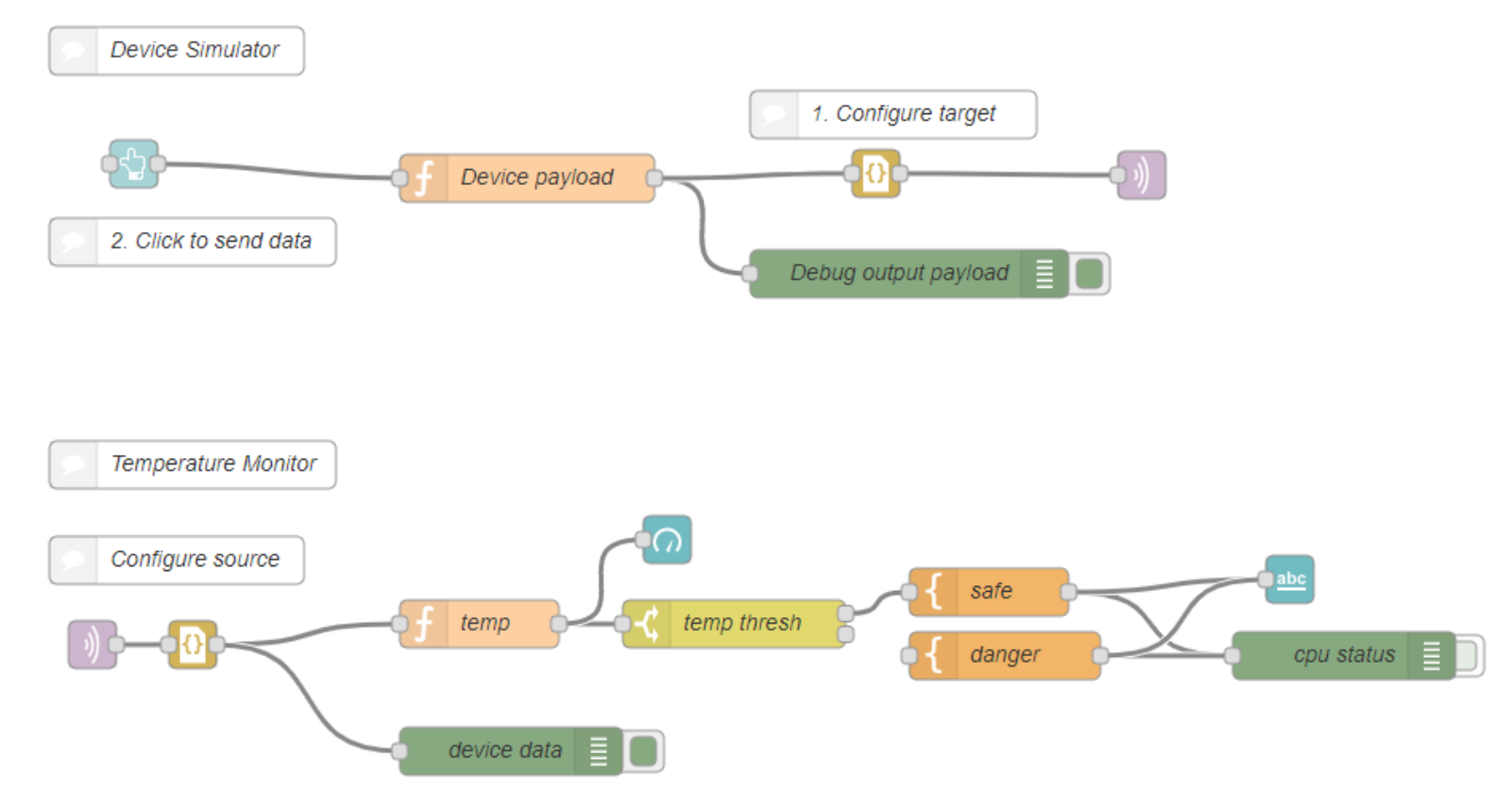
- Deploy your Node-Red instance. Once you go to the dashboard website, you should see you have different tabs to see – One should be the one using the IBM IoT communication channel (TAB1 or MainTab) and the other one should be using the MQTT broker (Tab2). **Press the button on the new one to make sure the system, and your MQTT channel, works!!!**



EDITING OUR EXAMPLE FURTHER TO SEND DATA

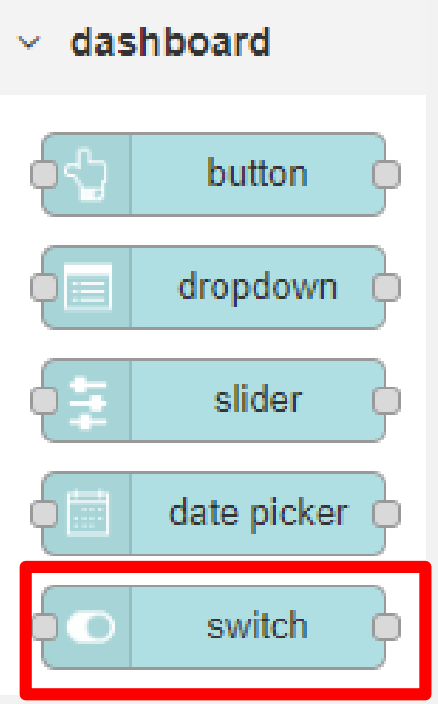
NODE-RED – USING OUR MQTT BROKER

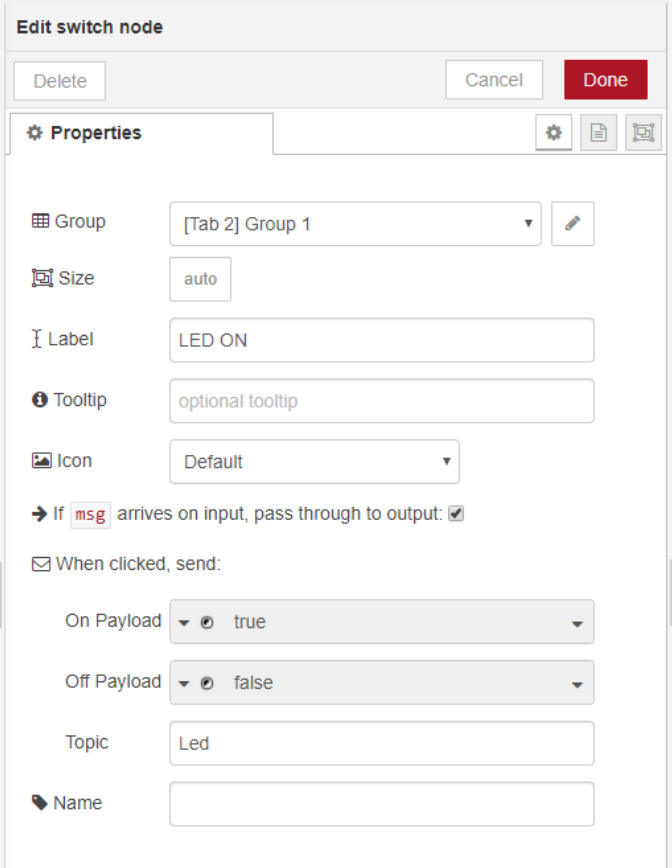
- Let's edit our example further to allow us to send data to our MQTT broker when we do an action. We will do this on the following slides.



NODE-RED – USING OUR MQTT BROKER

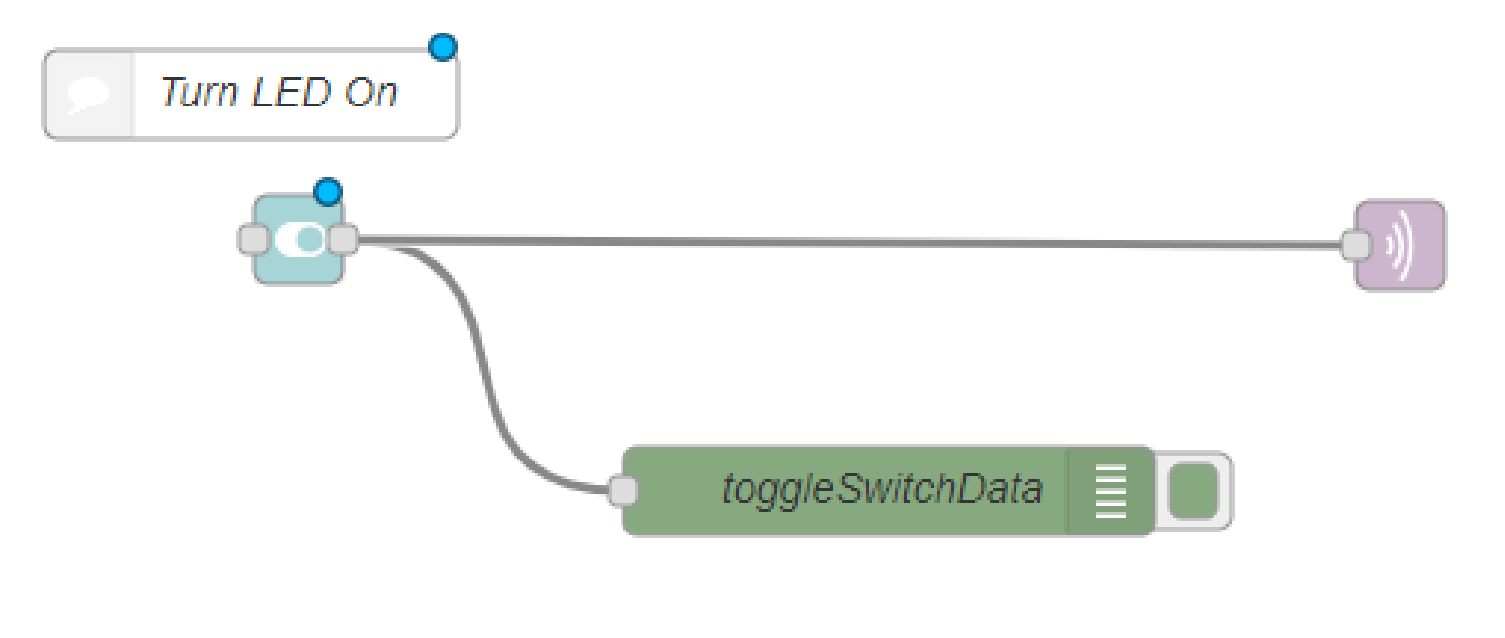
- Add a "switch" node from the dashboard section. You can select directly on the switch the Topic and data to send when the button is active or inactive. Double click on the button and add a Topic called "LedTopic". Also, add the switch to the 2nd Tab, since said tab is the one that uses the MQTT service rather than the IBM IoT service.





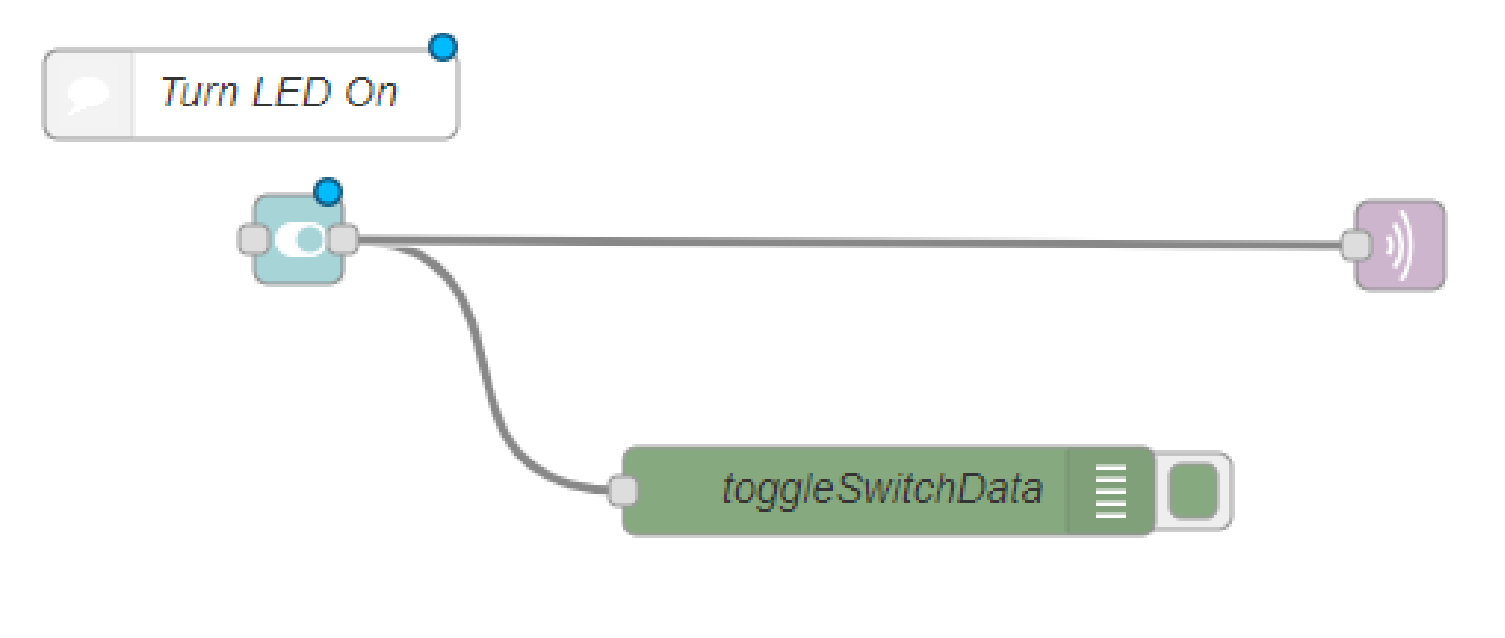
NODE-RED – USING OUR MQTT BROKER

- Now, add an MQTT output configured as before, alongside a debug node, as shown below.



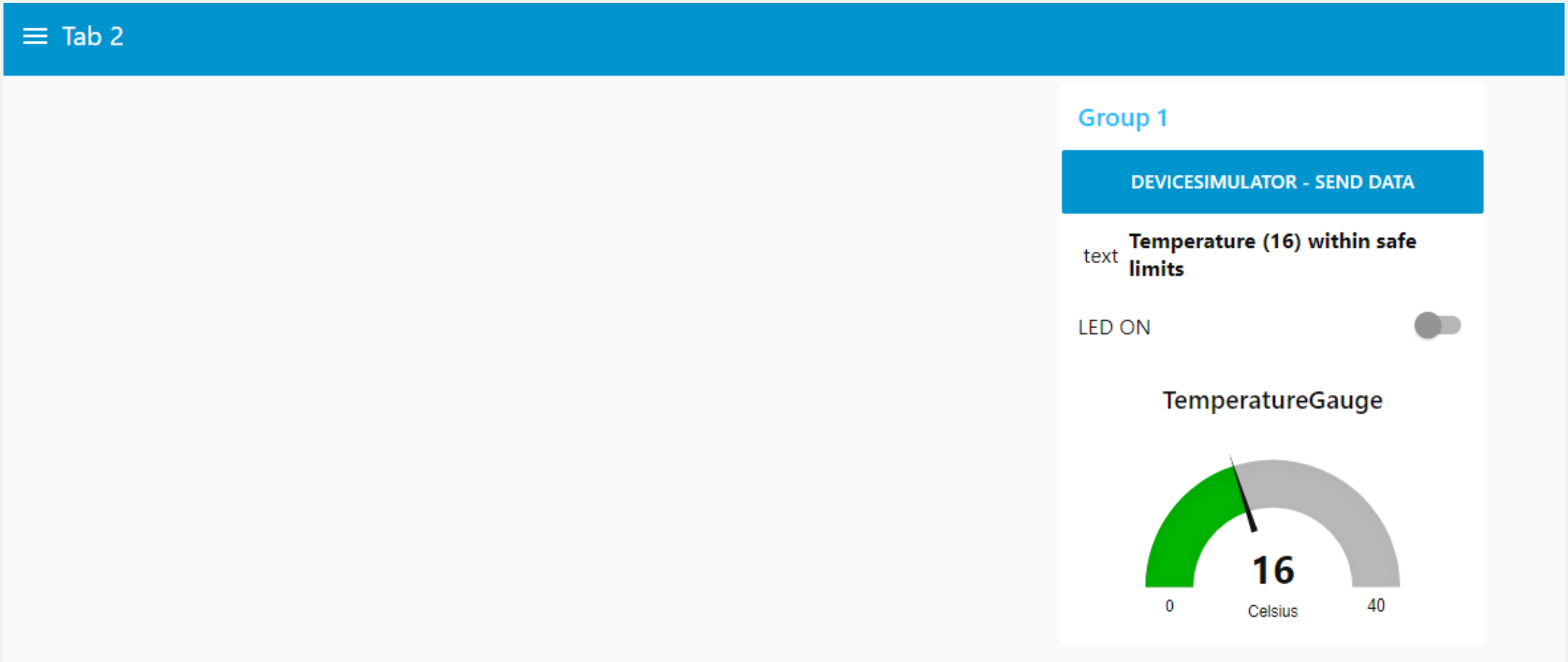
NODE-RED – USING OUR MQTT BROKER

- Now, add an MQTT output configured as before, alongside a debug node, as shown below. Later on we will use this button to turn an LED on the SAMW25 Xplained Board



NODE-RED – USING OUR MQTT BROKER

- **Deploy your solution.** Your solution should look like the following:



TRYING OUT THE STARTER CODE

TRYING OUT THE A7 STARTER CODE

- **Download the starter code from the a7 Google Drive** : https://drive.google.com/open?id=1e8MewsmgyNZBDivME_BWbfg7El75wrdt

- **Recommendation – Please don't work from the S Drive, sometimes it has some weird issues for which we cannot help. You will need to contact CETS directly for assistance.**

TRYING OUT THE A7 STARTER CODE

- **Change the SSID and password to the wifi channel you will use (main.h)**

```
/** Wi-Fi AP Settings. */
#define MAIN_WLAN_SSID                    "ChangeThisToo" /**< Destination SSID */
#define MAIN_WLAN_AUTH                    M2M_WIFI_SEC_WPA_PSK /**< Security manner */
#define MAIN_WLAN_PSK                     "Changethis" /**< Password for Destination SSID */

/** IP address parsing. */
```

- **Change the parameters of the MQTT broker to use your own MQTT broker (main.h)**

```
//Cloud MQTT User
#define CLOUDMQTT_USER_ID   "ChangeThis"


//Cloud MQTT pASSWORD
#define CLOUDMQTT_USER_PASSWORD "ChangeThisToo"

#define CLOUDMQTT_PORT      11625
```

TRYING OUT THE A7 STARTER CODE

- **Run the code. The code will**
- **1.) Download a PDF File (see main.h,** #define MAIN_HTTP_FILE_URL)
- **2.) Connect to the MQTT Server**
  - **Pressing the SW0 button will send the temperature. The temperature should increase on your Node Red Dashboard, Tab 2**
  - **Pressing the LED ON switch on the Node Red should turn on the LED on and off from the MCU**
  - **The terminal should show that everything worked.**

```
init_storage: please plug an SD/MMC card in slot...
init_storage: mounting SD card...
init_storage: SD card mount OK.
(APP)(INFO)Chip ID 1503a0
(APP)(INFO)DriverVerInfo: 0x13301354
(APP)(INFO)Firmware ver   : 19.5.4 Svnrev 15567
(APP)(INFO)Firmware Build Oct  4 2017 Time 14:59:09
(APP)(INFO)Firmware Min driver ver : 19.3.0
(APP)(INFO)Driver ver: 19.5.4
(APP)(INFO)Driver built at Apr  6 2019  21:43:11
main: connecting to WiFi AP Cisco12703...
wifi_cb: M2M_WIFI_CONNECTED
wifi_cb: IP address is 192.168.1.122
start_download: sending HTTP request...
(APP)(INFO)Socket 0 session ID = 1
resolve_cb: www.orimi.com IP address is 109.120.160.162

http_client_callback: HTTP client socket connected.
http_client_callback: request completed.
http_client_callback: received response 200 data size 20597
store_file_packet: creating file [0:pdf-test-037.pdf]
store_file_packet: received[1152], file size[20597]
store_file_packet: received[2598], file size[20597]
store_file_packet: received[4044], file size[20597]
store_file_packet: received[5490], file size[20597]
store_file_packet: received[6936], file size[20597]
store_file_packet: received[8382], file size[20597]
store_file_packet: received[9828], file size[20597]
store_file_packet: received[11274], file size[20597]
store_file_packet: received[12720], file size[20597]
store_file_packet: received[14166], file size[20597]
store_file_packet: received[15612], file size[20597]
store_file_packet: received[17058], file size[20597]
store_file_packet: received[18504], file size[20597]
store_file_packet: received[19950], file size[20597]
store_file_packet: received[20597], file size[20597]
store_file_packet: file downloaded successfully.
main: please unplug the SD/MMC card.
main: done.
(APP)(INFO)Socket 1 session ID = 1

Connecting to Broker...MQTT Connected
MQTT Connected to broker
```

RECOMMENDATIONS FOR A7

- **Step through the code! Make sure you understand what is going on on each step.**
- **Step through the example and investigate Node-Red. Visualize how your final product will be and what it will use.**
- **Architect your code before diving in to doing code**